目　录

第 1 章　MATLAB 6 概述

MATLAB 是近几年来在国外广泛流行的一种可视化科学计算软件，它不但具有语法结构简单、数值计算高效、图形功能完备和图像处理方便的特点，还具有开发符号计算、文字处理、可视化建模仿真和实时控制的能力，该软件已成为适合多学科、多部门要求的新一代科技应用软件。

本章首先简单介绍 MATLAB 软件的发展历史、主要特点和安装过程，然后对 MATLAB 的界面环境进行重点介绍。

1.1　MATLAB 简介

MATLAB 语言是由美国 MathWorks 公司推出的计算机软件，经过多年的逐步发展与不断完善，现已成为国际公认的最优秀的科学计算与数学应用软件之一。其内容涉及矩阵代数、微积分、应用数学、有限元法、科学计算、信号与系统、神经网络、小波分析及其应用、数字图像处理、计算机图形学、电子线路、电机学、自动控制与通信技术、物理、力学和机械振动等方面。MATLAB 的特点是语法结构简单，数值计算高效，图形功能完备，特别受到以完成数据处理与图形图像生成为主要目的的科研人员的青睐。各国的高校学生(包括硕士生与博士生)也将 MATLAB 作为必须掌握的基本程序设计语言。

1.1.1　MATLAB 的发展简史

MATLAB 是 Matrix Laboratory(矩阵实验室)的缩写，最初由美国 Cleve Moler 博士在 70 年代末讲授矩阵理论和数据分析等课程时编写的软件包 Linpack 与 Eispack 组成，旨在使应用人员免去大量经常重复的矩阵运算和基本数学运算等繁琐的编程工作。1984 年，Cleve Moler 博士和一批数学家、软件专家组建了 MathWorks 公司，开发出了第 2 代 MATLAB 软件，并推向市场。其内核改用 C 语言编写，提高了速度，另外还增加了绘图功能，使数值计算结果可以直接在 MATLAB 环境下用曲线和曲面等形式表示出来。1990 年，MathWorks 公司推出了以框图为基础的控制系统仿真工具 Simulink，它方便了系统的研究和开发，使控制工程师可以直接构造系统框图进行仿真，并提供了控制系统中常用的各种环节的模块库。1993 年，MathWorks 公司推出的 MATLAB 4.0 版在原来的基础上又作了较大改进，并推出了 Windows 版，使命令执行和图形绘制可以在不同窗口进行。1994 年推出了 MATLAB 4.2 版，并得到了广泛的重视和应用。1999 年 1 月推出了 MATLAB 5.3 (Release 11.0)版本，真正实现了 32 位运算，其速度更快、功能更完善、界面更友好，并且提供了

Internet 搜索引擎，可协助用户寻求在线帮助。新版本 6.0(Release 12.0)、6.1 (Release 12.1) 又作了更精细的改进，6.1 版于 2001 年 5 月推向市场。相对于 5.3 版本，新版本增加了许多新的功能，主要表现在以下几个方面：

- 增强了用户界面的交互性，其窗口界面更加友好。
- 增加了工具箱的种类，增强了工具箱的功能。
- 增加了许多功能函数。
- 扩充了绘图功能。
- 增强了对多维矩阵，稀疏矩阵的运算功能。
- 增加了微分方程的解法，增加了积分方程的算法。
- 扩充了矢量和矩阵的类型。
- 提供了更新的和完备的在线帮助文档。
- 提供了输入数据向导(Import Wizard)。

1.1.2 MATLAB 的特点

MATLAB 之所以成为世界流行的科学计算与数学应用软件，是因为它有着下列强大的功能。

- 高质量、强大的数值计算功能。为满足复杂科学计算任务的需要，MATLAB 汇集了大量常用的科学和工程计算算法，从各种函数到复杂运算，包括矩阵求逆、矩阵特征值、奇异值、工程计算函数以及快速傅立叶变换等。MATLAB 强大的数值计算功能是其优于其他数学应用软件的重要原因。尤其是当今流行的 MATLAB 6 版本，其数值计算功能更加完善。
- 数据分析和科学计算可视化功能。MATLAB 不但科学计算功能强大，而且在数值计算结果的分析和数据可视化方面也远远优于其他同类软件。在科学计算和工程应用中，经常需要分析大量的原始数据和数值计算结果，MATLAB 能将这些数据以图形的方式显示出来，使数据间的关系清晰明了。
- 强大的符号计算功能。科学计算有数值计算与符号计算两种，在数学、应用科学和工程计算领域，常常会遇到符号计算问题，仅有优异的数值计算功能并不能解决科学计算时的全部需要。在 MATLAB 的发展过程中，MathWorks 公司从 Waterloo 大学购买了 Maple 的使用权，并以 Maple 的核心部分作为其符号计算功能的引擎，依靠 Maple 已有的库函数，实现了 MATLAB 环境下符号计算功能。
- 强大的非线性动态系统建模和仿真功能。MATLAB 提供了一个模拟动态系统的交互式程序 Simulink，允许用户通过绘制框图来模拟一个系统，并动态地控制该系统。Simulink 能处理线性、非线性、连续、离散等多种系统，它包括应用程序扩展集 Simulink Extensions 和 Blocksets。其中 Simulink Extensions 是支持在 Simulink 环境下进行系统开发的一些工具类应用程序，如 Simulink Accelerator、Real.Time Workshop 及 Stateflow；而 Blocksets 则是针对 DSP(数字信号处理)、Communications(通信)、Nonlinear Control Design(非线性控制设计)、Fixed Point(定点)等几个特殊应用领域设计的程序的集合。

- 灵活的程序接口功能。应用程序接口(API)是一个允许用户编写的与 MATLAB 互相配合的 C 或 Fortran 程序的文件库。MATLAB 提供了方便的应用程序接口 API，用户可以在 MATLAB 环境下直接调用已经编译过的 C 和 Fortran 子程序，在 MATLAB 和其他应用程序之间建立客户机/服务器关系。同样，在 C 和 Fortran 程序中，也可以调用 MATLAB 的函数或命令，使得这些语言可以充分利用 MATLAB 的矩阵运算功能和方便的绘图功能。
- 文字处理功能。MATLAB 记事本成功地将 MATLAB 与文字处理系统 Microsoft Word 集成为一个整体，为用户进行文字处理、科学计算、工程设计创造了一个统一的工作环境。用户不仅可以利用 Word 的文字编辑处理功能，方便地创建 MATLAB 的系统手册、技术报告、命令序列、函数程序、注释文档以及与 MATLAB 有关的教科书等 6 种文档，而且还能从 Word 访问 MATLAB 的数值计算和可视化结果。

另外，MATLAB 还具有支持科学计算标准的开放式可扩充结构和跨平台兼容的特点，能够很好地解决科学和工程领域内的复杂问题。

MATLAB 的技术特点主要表现在以下几个方面：

- 界面友好，编程效率高。MATLAB 是一种以矩阵为基本变量单元的可视化程序设计语言，语法结构简单，数据类型单一，命令表达方式接近于常用的数学公式。这使 MATLAB 用户在短时间内就能快速掌握其主要内容和基本操作。MATLAB 不仅能免去大量的经常重复的基本数学运算，而且其编译和执行速度都远远超过了采用 C 和 Fortran 语言设计的程序。可以说，MATLAB 在科学计算与工程应用方面的编程效率远远高于其他高级语言。
- 功能强大，可扩展性强。MATLAB 语言不但提供了科学计算、数据分析与可视化、系统仿真等强大的功能，而且还具有可扩展性特征。MathWorks 公司针对不同领域的应用，推出了自动控制、信号处理、图像处理、模糊逻辑、神经网络、小波分析、通信、最优化、数理统计、偏微分方程、财政金融等 30 多个具有专门功能的 MATLAB 工具箱。各种工具箱中的函数可以互相调用，也可以由用户更改。MATLAB 支持用户对其函数进行二次开发，用户的应用程序可以作为新的函数添加到相应的工具箱中。
- 图形功能灵活方便。MATLAB 具有灵活的二维与三维绘图功能，在程序的运行过程中，您可以方便迅速地用图形、图像、声音、动画等多媒体技术直接表述数值计算结果，可以选择不同的坐标系，可以设置颜色、线型、视角等，还可以在图中加上比例尺、标题等标记，在程序运行结束后改变图形标记、控制图形句柄等，并且还可以将图形嵌入到用户的 Word 文件中。
- 在线帮助，有利于自学。用户可以借助于 MATLAB 环境下的“在线帮助”学习各种函数的用法及其内涵。对于 MATLAB 5.x 以上版本，还可以用 HTML 方式查询更为详细的参考资料。另外还可以直接访问 MathWorks 公司的网站，以获得常见问题解答(FAQ)、产品指南和 MATLAB 书籍等更丰富的帮助信息。

总之，MATLAB 语言已经成为科学计算、系统仿真、信号与图像处理的主流软件。

1.2 安装 MATLAB

安装 MATLAB 的具体要求如下：

- CPU：Pentium、Pentium Pro、Pentium II、Pentium III、Pentium 4 或 AMD Athlon。
- 操作系统：Microsoft Windows 95、Windows 98、Windows NT 4.0 或 Windows 2000。
- 内存：至少 64 MB，建议 128 MB。
- 硬盘：根据安装程序提示决定，完全安装需要 750 MB 左右。
- 显示卡：至少 8 位，即 256 色。
- 显示器：至少能支持 256 色，分辨率为 800×600 像素。
- 安装时需光驱。

MATLAB 的安装比较简单，下面以在 Windows 98/2000 下安装 MATLAB 6 为例进行介绍。

(1) 启动安装程序。

放入 MATLAB 6 的安装盘，安装程序将自动运行，或者双击安装盘中的 setup 文件，安装程序将显示如图 1.1 所示的安装程序界面，稍后出现如图 1.2 所示的对话框，单击 Next 按钮进入下一步。

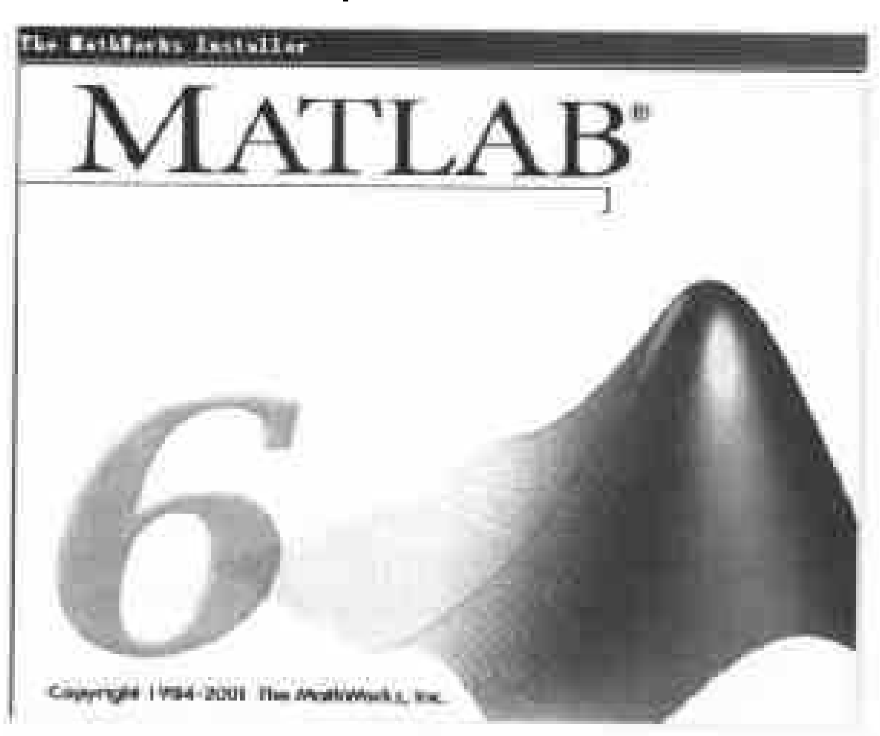


图 1.1 MATLAB 6安装程序界面

(2) 输入个人许可密码(Personal License Password，PLP)。

在图 1.3 中填写许可密码，此密码(又称序列号)应该随着购买的MATLAB 软件一起提供。单击 Next 按钮进入下一步。

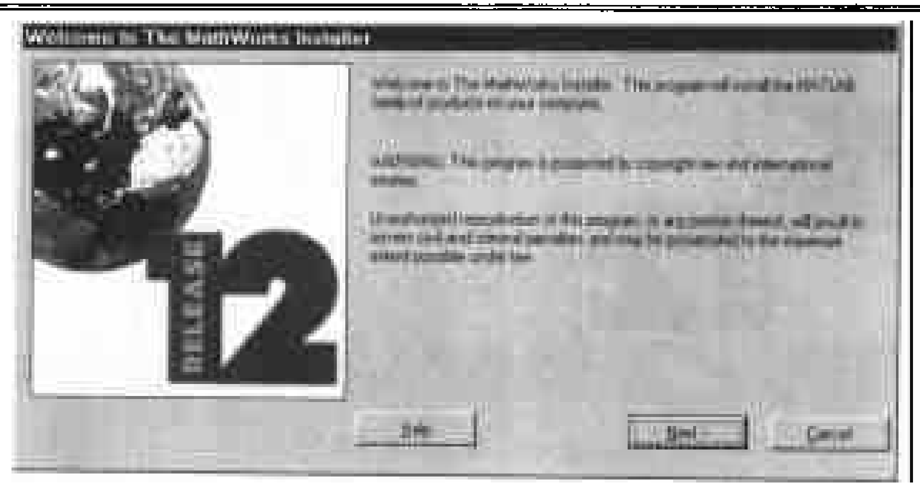


图 1.2 进入MATLAB安装程序

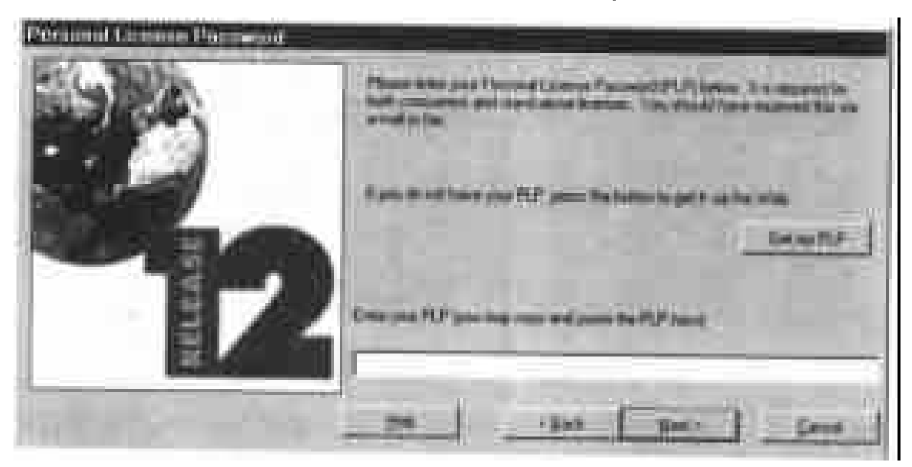


图 1.3 输入软件协议密码

(3) 浏览软件使用许可协议。

在如图 1.4 所示的对话框中，单击 Yes 按钮接受此协议进入下一步。

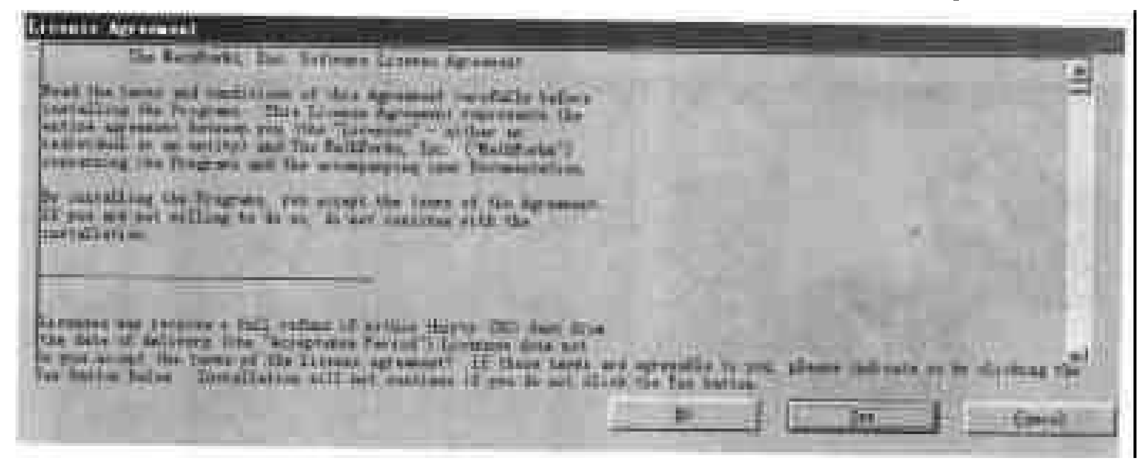


图 1.4 软件使用许可协议对话框

(4) 输入用户信息。

系统弹出如图 1.5 所示的对话框，输入用户姓名和单位名称后，单击 Next 按钮进入下一步。

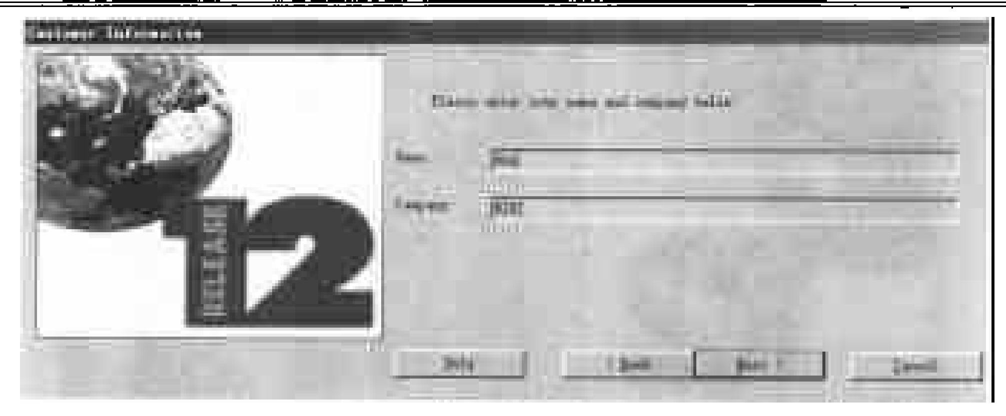


图 1.5 输入用户信息

(5) 选择 MATLAB 安装组件。

系统会弹出两个对话框，第一个对话框如图 1.6 所示，询问用户是否下载最新软件更新，如需下载可单击 Yes 按钮，直接安装可单击 No 按钮。



图 1.6 是否下载最新软件提示对话框

在如图 1.7 所示的对话框中，选择安装目录、安装的组件，其中有各种可选的工具箱及其帮助文件(PDF 或 HTML)，Image Toolbox 是图像处理工具箱，根据需要选择所要安装的工具箱。如果用户的计算机上安装了 Acrobat Reader 软件，则可选择 PDF 类型的帮助文件；如果用户的计算机上安装了浏览器，则可选择 HTML 类型的帮助文件。对话框右端有安装所需空间提示，注意硬盘空间一定要大于该提示数字。单击 Next 按钮开始安装。

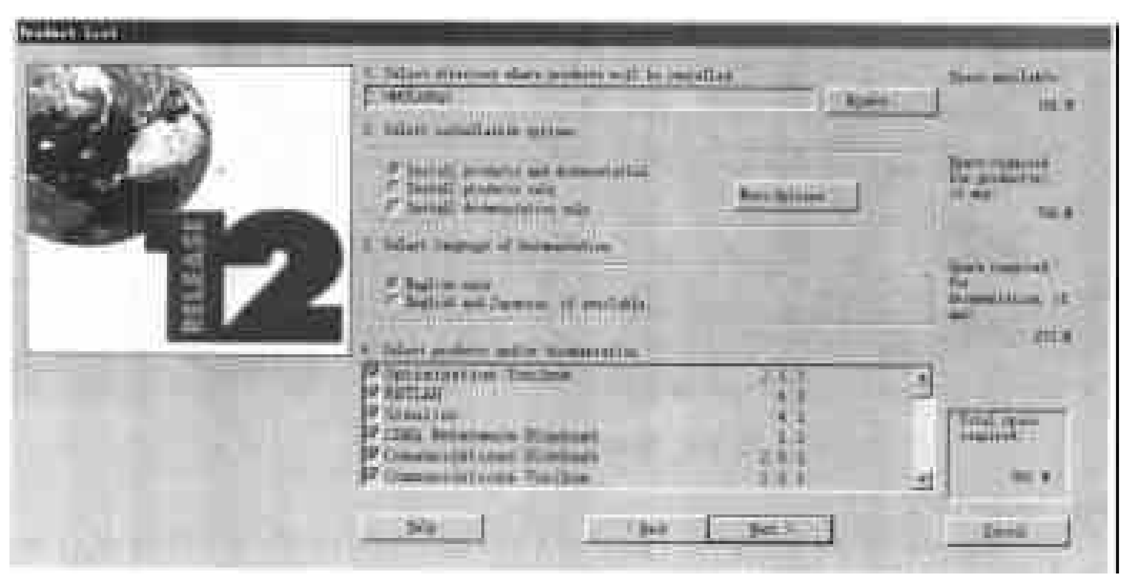


图 1.7 Product List对话框

显示如图 1.8 所示的安装状态窗口，由于选择的内容不同，安装过程中可能会与下图显示的状态不同。接下来按照提示插入第 2 张光盘(文档安装盘)。

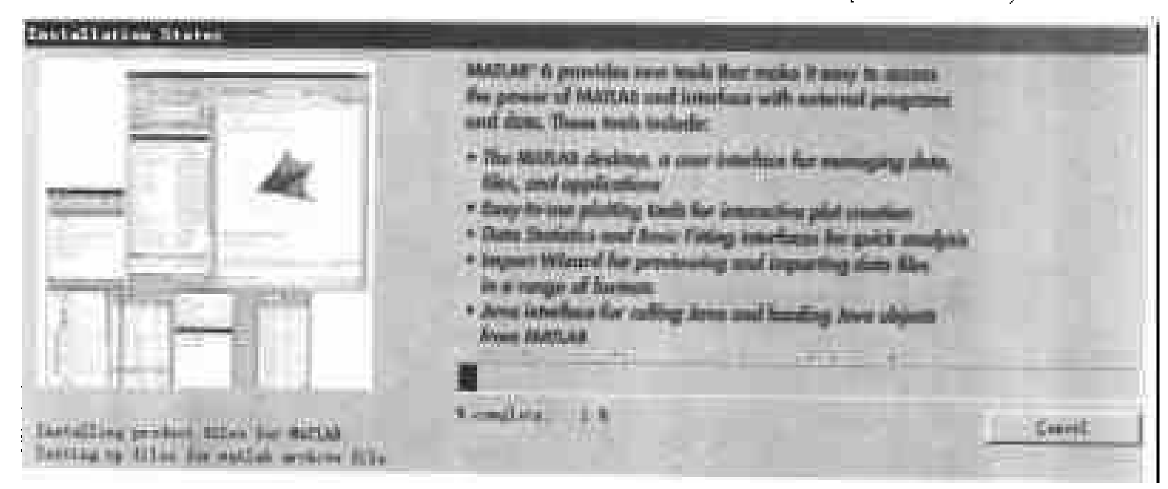


图 1.8 安装状态窗口

(6) 完成安装。

复制完文件后系统会弹出如图 1.9 所示的对话框，单击 Finish 按钮完成安装。选择第 1 个单选按钮，并重新启动系统后，MATLAB 6 就可以运行了。如果选择第 2 个单选按钮，即可立即运行 MATLAB 6。

在安装过程中，如果出现问题，可单击 Help 按钮，获得相应的帮助。

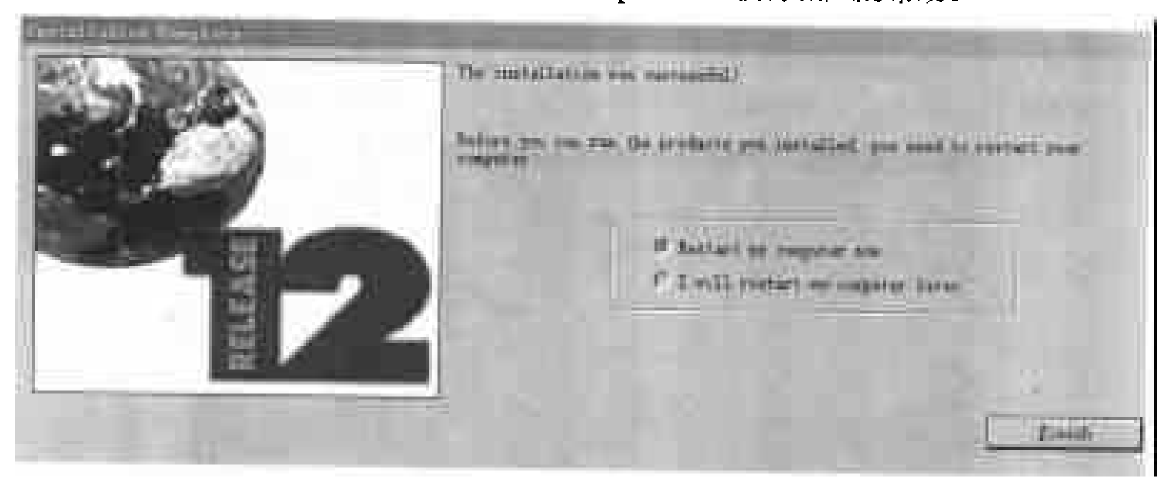


图 1.9 Installation Complete对话框

1.3 MATLAB 的界面环境

与以前版本相比，MATLAB 6 增强了用户界面的交互性，有了全新的桌面视窗环境。启动 MATLAB 6 后，显示的界面窗口如图 1.10 所示。在主窗口中，层叠平铺了 Command Window(命令窗口)、Launch Pad(发射台)、Workspace(工作空间)、Command History (命令历史记录)、Current Directory (当前目录)等 6 个子窗口。

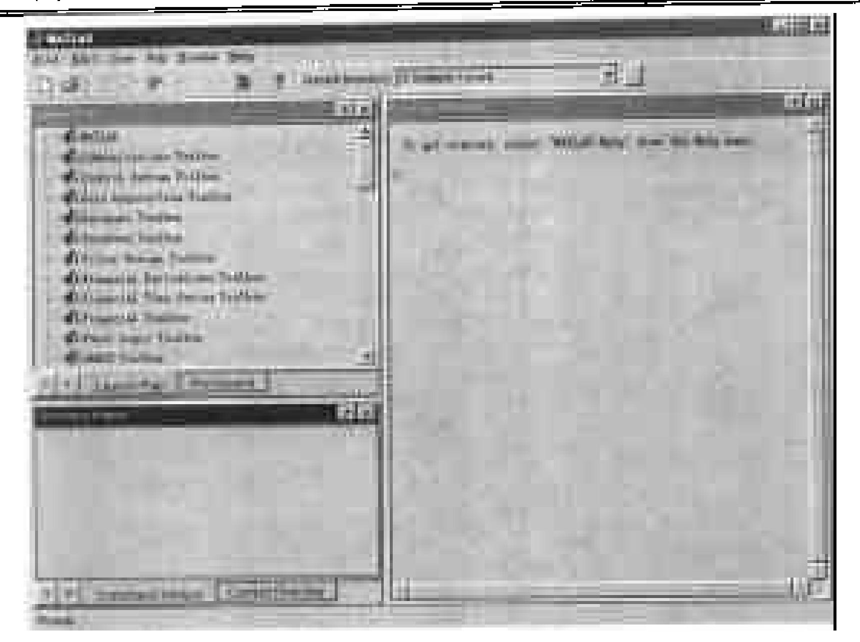


图 1.10 MATLAB 6界面图

1.3.1 Command Window 窗口

Command Window 窗口是 MATLAB 界面中的重要组成部分，利用这个窗口可以和 MATLAB 进行交互操作，即输入数据或命令并进行相应的运算，单击窗口标题栏中的按钮可以单独打开 Command Window 窗口，如图 1.11 所示。启动该子窗口后，窗口第一行提示可选择 MATLAB Help 获得帮助。下面在窗口中进行一些基本运算。

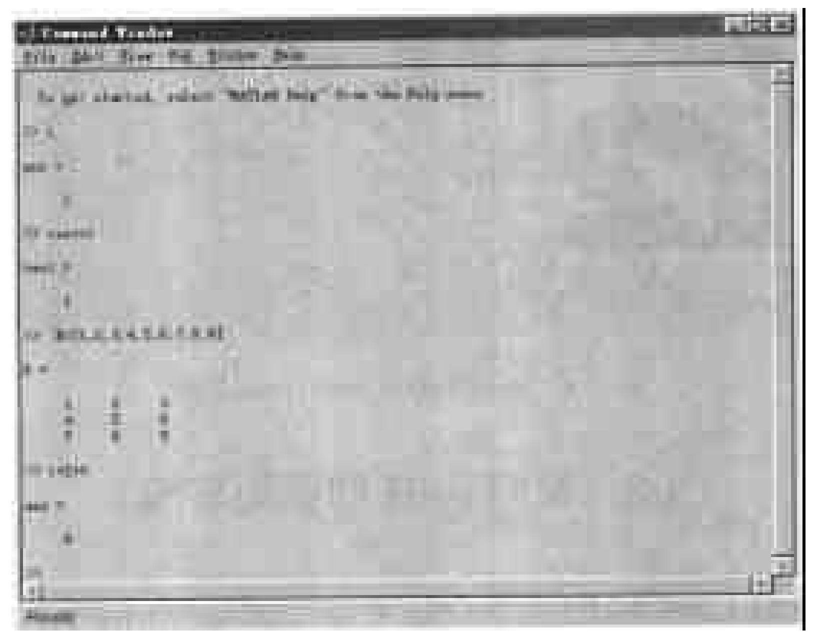


图 1.11 Command Window窗口

在提示符>>后输入数字“1”，按 Enter 键，窗口中将显示如下：

```
ans =
1
```

ans 是结果的默认变量名。

继续输入命令“test=1”，按 Enter 键换行，显示如下结果：

```
test =
1
```

系统解释此命令为给变量 test 赋值 1。

若输入“B=[1,2,3;4,5,6;7,8,9]”，系统解释此命令为赋予 B 一个 3×3 数组，按 Enter 键执行显示结果如下：

```
B =
     1     2     3
     4     5     6
     7     8     9
```

在 Command Window 窗口中也可输入表达式，如“1+2+3”，按 Enter 键后，将运算值赋予结果默认变量 ans，显示结果如下：

```
ans =
     6
```

在 Command Window 窗口中还可以输入许多命令，本章后续部分将会重点介绍。

在 Command Window 窗口中也可以对已输入的命令进行编辑，表 1.1 列出了控制光标位置及对命令进行操作的一些常用快捷键。

表 1.1 命令快捷键和功能捷键

功能键	快捷键	功能说明
↑	Ctrl+p	调出前一个命令行
↓	Ctrl+n	调出后一个命令行
←	Ctrl+b	光标左移一个字符
→	Ctrl+f	光标右移一个字符
Ctrl+←	Ctrl+l	光标左移一个单词
Ctrl+→	Ctrl+r	光标右移一个单词
Home	Ctrl+a	光标移至行首
End	Ctrl+e	光标移至行尾
Esc	Ctrl+u	清除当前行
Del	Ctrl+d	清除光标所在位置后的字符
Backspace	Ctrl+h	清除光标所在位置前的字符
	Ctrl+k	删除到行尾
	Ctrl+c	中断正在执行的命令

1. Command Window窗口菜单

Command Window 窗口是一个标准的 Windows 界面，利用菜单中的命令可以完成对工作窗口的操作。它的使用方法与一般的应用程序相同。

File、Edit 及 View 菜单的命令如图 1.12、图 1.13 及图 1.14 所示，其意义如表 1.2 所示。

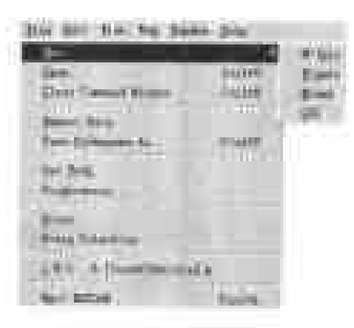


图 1.12 File菜单命令



图 1.13 Edit菜单命令

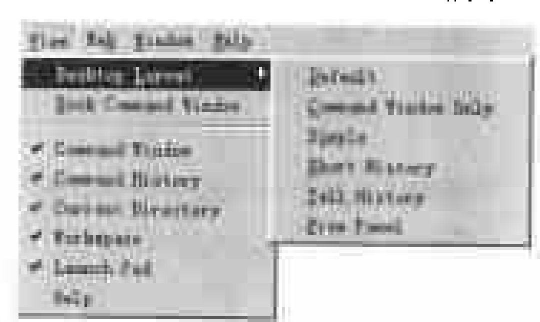


图 1.14 View菜单命令

表 1.2 File菜单

菜单命令	功　能
File 菜单	
New	新建一个 M 文件、图形或 Simulink 模块
Open	打开一个已有的文件，可以是 M 文件、图形、Simulink 模块
Close Command Window	关闭 Command Window 子窗口
Import	导入外部文件数据
Save Workspace As	将当前工作空间另存为其他文件
Set Path	设置 MATLAB 搜索路径
Preferences	设置 MATLAB 的工作环境参数
Print	打印输出
Print Selection	打印所选中的文本或其他对象
Exit MATLAB	退出 MATLAB 系统
Edit 命令	
Undo	取消上一步操作
Redo	重新执行上一步操作
Cut/Copy/Paste	删除/复制/粘贴
Paste Special	选择性粘贴
Select All	全部选择
Delete	删除对象
Clear Command Window	清除 Command Window 窗口中的内容
Clear Command History	清除 Command History 窗口中的内容
Clear Workspace	清空工作空间
View 命令	
Desktop Layout	设置窗口的布置方式
Dock Command Window	将 Command Window 窗口停靠在 MATLAB 主窗口

续表

菜单命令	功　能
Command Window/Command History/Current Directory/Workspace/Launch Pad	进行窗口切换
Help	启动帮助窗口

2. Command Window窗口的设置

在 MATLAB 主窗口选择 File | Preferences 命令，可以在打开的 Preferences 对话框中设置 MATLAB 的工作环境参数。选择 Command Window 选项，则可以设置 Command Window 窗口的参数，如图 1.15 所示。

- Text display 选项组

  该选项组用来设置 Command Window 窗口的文本显示格式。

  - Numeric format 下拉列表框用来设置 Command Window 窗口的数值显示格式，格式说明如表 1.3 所示。
  - Numeric display 下拉列表框用来控制 Command Window 窗口的数据输出风格，选择 loose 选项时，将在显示结果中加入一些空行；选择 compact 选项将压缩掉这些空行。

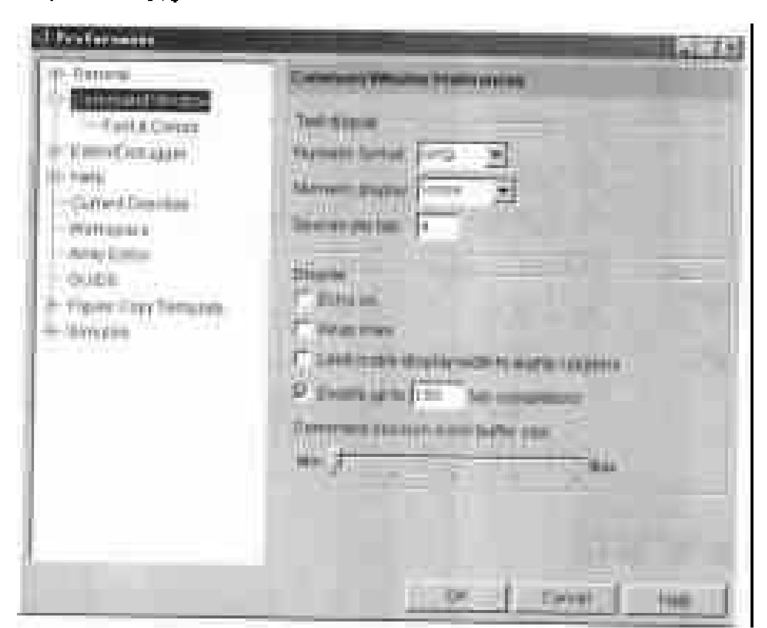


图 1.15 Command Window窗口的Preferences对话框

表 1.3 Command Window窗口数值显示格式

显示格式	说　明
short	5 位定点表示
short E	5 位浮点表示
short G	系统选择 short 和 short E 中最好的表示

续表

显示格式	说　明
long	15 位定点表示
long E	15 位浮点表示
long G	系统选择 long 和 long E 中最好的表示
bank	用元、角、分(美制)定点表示
+	仅显示数值的符号
ration	近似的有理数表示
hex	十六进制表示

- Display 选项组

  Display 选项组用来选择运行 MATLAB 程序时是否在 Command Window 窗口中显示正在运行的命令，是否自动换行，是否限制矩阵的显示宽度为 80 列，还可以设置 Command Window 窗口中命令文本显示的缓存大小等。

另外，Command Window 窗口的字体和颜色设置参看 1.4.6 节有关 Editor/Debugger 参数的设置内容。

1.3.2 Launch Pad 窗口



图 1.16 Launch Pad 窗口

用户可以在 Launch Pad 窗口中启动某个工具箱的应用程序，单击 Launch Pad 窗口中的按钮后，Launch Pad 窗口就最大化，如图 1.16 所示。通过 Launch Pad 窗口，可以打开各个工具箱的帮助、Demos(演示窗口) 和其他相关的文件或应用程序，这是一个非常好的工具。通过它，用户可以很方便地从事自己的工作，比如要启动 Image Processing Toolbox(图像处理工具箱)的 Demos，双击该项目即可。

Launch Pad 窗口的命令与 Command Window 窗口一样，具体内容参看 1.3.1 节。

1.3.3 Workspace 窗口

旧版本的 Workspace 是一个对话框，可操作性差，6.0 版后的 Workspace 作为一个独立的窗口，如图 1.17 所示。单击 Workspace 窗口的按钮后，工作空间就最大化。

1. Workspace窗口菜单

Workspace 窗口的命令与 Command Window 窗口类似，只是在 View 菜单下增加了 Workspace View Options 命令，如图 1.18 所示。





图 1.17 Workspace窗口



图 1.18 Workspace View Options 菜单

该命令用于设置 Workspace 窗口中变量的显示属性，各命令的主要功能如表 1.4 所示。

表 1.4 变量的显示属性

菜单命令	功　能	菜单命令	功　能
Show Size	显示大小	Sort by Size	按大小排序
Show Bytes	显示字节数	Sort by Bytes	按字节数排序
Show Class	显示数据类型	Sort by Class	按数据类型排序
Sort by Name	按名称排序	Sort Ascending	按升序排列

2. 查看与清除工作空间中的变量

MATLAB 的工作空间包含一组变量(或命名的数组)，用户可以在 Command Window 隔窗口对这些变量进行操作。用户可以用 who(必须小写，MATLAB 系统是识别大小写的，对于变量和命令都是这样)或 whos(必须小写)命令查看当前工作空间的内容。who 命令可以给出简明的变量名列表，而 whos 命令则还可以列出变量的大小及数据类型。如图 1.19 所示为用 who 命令列出工作空间中的所有变量；图 1.20 为用 whos 列出当前工作空间的变量。



图 1.19 用who命令列出工作空间中的变量



图 1.20 用whos命令列出工作空间中的变量大小和类型

选择 Edit | Clear+[窗口]命令可以清除窗口中的内容，比如，选择 Edit | Clear Command Window 命令，可删除命令窗口的所有显示。

如果选择 Edit | Clear Workspace 命令，则清除所有变量，选择该命令后，系统会弹出一个提示对话框，让用户确认是否清除，因为清除了就无法恢复。

通过工作空间浏览器(即 Workspace 窗口)也可以直接查看当前工作空间中的所有变量，如图 1.17 所示。其中列出了变量名称、大小、字节数和变量类型，与 whos 命令所列出的信息相同。选择其中一个变量，按 Delete 键可删除这个变量，也可以单击按钮进行操作，或者选择 Edit | Delete 命令。

3. 数组内容的显示和编辑

双击工作空间浏览器中的变量，如二维数组 B，系统会弹出 Array Editor(数组编辑器)窗口，也可以单击按钮。Array Editor 窗口中显示了二维数组 B 的内容，如图 1.21 所示。用户可以在 Array Editor 窗口中对数组的大小及每个元素进行编辑，但不可以改变数组的类型，例如，在数值型数组中输入字符串，系统就会自动给出错误提示。

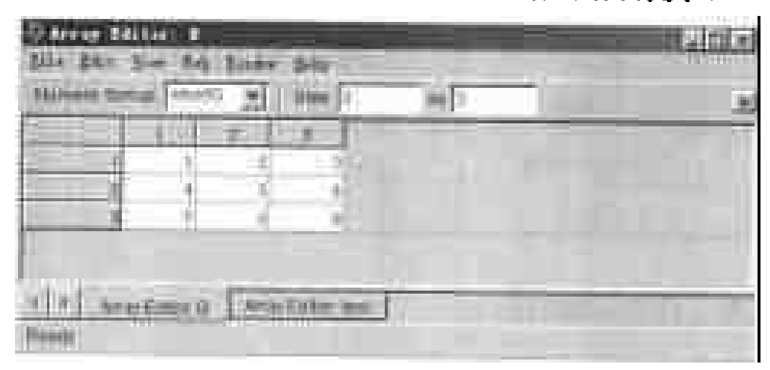


图 1.21 Array Editor窗口

4. 保存工作空间

MATLAB 允许用户保存当前的工作空间，随时调用已保存的工作空间，并可进行文本数据文件的输入和输出。

使用 save 命令把工作空间保存在二进制的 MAT 文件中，以后可以使用 load 命令调用 MAT 文件。保存整个工作空间的内容，例如，在 Command Window 窗口中使用以下命令：

```
save worksp,
```

把当前工作空间的内容保存在当前目录下的 worksp1.mat 文件中。

如果仅要保存工作空间中的几个变量，可以在保存的文件名后加上要保存的变量列表，例如，把当前工作空间中的变量 var1,var2 保存在 worksp2.mat 文件中，可以使用以下命令：

```
save worksp2 var1 var2
```

另外，选择 File | Save Workspace As 命令，也可以保存工作空间。

在保存工作空间时可以指定一些参数，控制保存文件的格式。各参数如表 1.5 所示。

表 1.5 保存工作空间的参数

参 数	作 用
.mat	使用二进制 MAT 文件格式(系统默认的格式)
.ascii	使用 8 位 ASCII 格式
.ascii .double	使用 16 位 ASCII 格式
.ascii .double .tabs	使用制表符分隔数组元素
.v4	保存为与 MATLAB 4 兼容的格式
.append	向已经存在的 MAT 文件中添加数据

例如，要把当前工作空间中的所有变量以 ASCII 格式保存到 data1 文件中，可以在 Command Window 窗口中输入如下命令：

```
save data1.ascii
```

注意： 使用参数 v4 时只能保存 MATLAB 6 中与 MATLAB 4.x 兼容的数据类型，因此像结构体、单元数组和对象不被保存，而且保存的文件名应为 MATLAB 4.x 支持的文件名格式。使用 ASCII 格式保存工作空间内容时，每次仅保存一个变量。如果多于一个变量，仍可以创建 ASCII 保存变量，但是不能使用 load 命令加载已保存的数据。

5. 加载工作空间

用 save 命令保存的工作空间文件，可以使用 load 命令加载。例如：

```
load worksp
```

如果当前工作空间与加载的工作空间具有相同的变量，则后加载的变量将自动覆盖与原工作空间中同名的变量。

如果保存 MAT 文件使用了其他扩展名，如“worksp.s”。加载时则要使用参数“.mat”指定其为 MAT 格式的文件，并且一定要加上文件的扩展名。例如：

```
load worksp.s.mat
```

在主窗口、Command Window 窗口和 Workspace 窗口中选择 File | Import Data 命令，也可以加载工作空间。加载时系统会弹出加载工作空间向导，如图 1.22 所示。

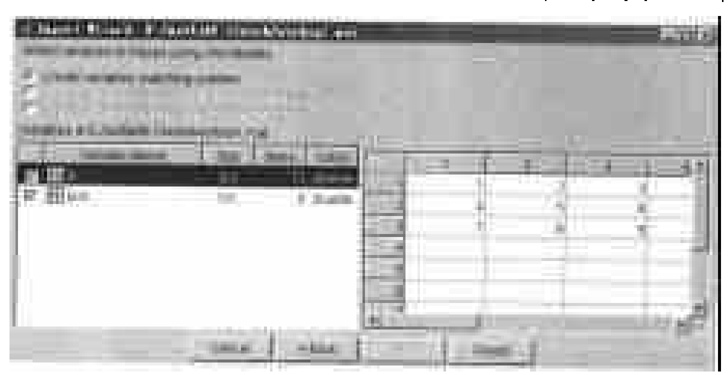


图 1.22 加载工作空间向导

通过此向导，可以从 MAT 数据文件中选择加载到工作空间中的变量。

使用 load 命令也可以加载 ASCII 数据文件，把文件的内容读取到一个与文件名相同的变量中，如图 1.23 所示。

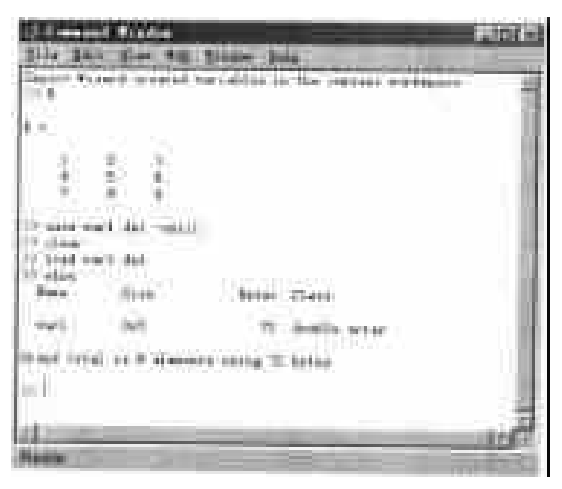


图 1.23 加载ASCII数据文件

图 1.23 中的各命令的意义如表 1.6 所示。

表 1.6 Command Window窗口中的命令及意义

命 令	意 义
save var1.dat –ascii	以 8 位格式保存当前的工作空间
clear	清除当前工作中的所有变量
load var1.dat	加载已保存的 ASCII 格式
whos	列出当前工作空间中的变量

从图 1.23 的结果中可以看出，以前定义的变量 B，不在当前的工作空间中，而当前工作空间中的变量名为 var1，是保存工作空间的文件名。此变量的大小、类型和字节数与变量 B 完全一样。

6. 文件名和变量名的特殊用法

如果文件名和变量名保存在字符串变量中，在保存或加载工作空间时，则可以使用字符串变量，同时可以把 save 和 load 命令理解为一般的函数，参数的顺序与命令中参数的顺序相同。例如下面的一段命令：

```
a=1
b=1
save('myfile','a','b')
A='myfile'
Load(A)    %使用字符型变量 A 作为函数 load 的参数。
```

等效于以下命令：

```
save myfile a b
load myfile
```

同样也等效于以下命令：

```
A='myfile';
B='a';
C='b';      %定义 3 个字符变量保存文件名和变量名。
Save(A,B,C);   %使用 save 命令的函数方式，保存字符变量 A,B 和 C。
load(A)
```

命令 save 和 load 都可以使用通配符"*"搜索相似的变量名。例如：

```
save myworksp a*
```

把当前工作空间中所有以字符 a 开头的变量保存在文件 myworksp.mat 中。
又如：

```
load myworksp aa*bb
```

从文件中 myworksp.mat 加载所有变量名中前两个字符是 aa 的变量，最后两个字符是 bb 的变量。通配符"*"的位置可以是任意字符或字符串。

1.3.4 Command History 窗口

Command History 窗口主要显示已执行过的命令。MATLAB 每次启动时，Command History 窗口会自动记录启动的时间，并将 Command Window 窗口中执行的命令记录下来。一方面便于查找，另一方面可以再次调用这些命令，如图 1.24 所示。

双击 Command History 窗口中的三维数组 B，该操作等效于在 Command Windows 窗口中输入此命令，如图 1.25 所示。



图 1.24 调用Command History窗口中的命令　　图 1.25 执行Command History窗口中的命令

Command History 窗口的菜单与 Command Window 窗口中的菜单完全一样，参看 1.3.1 节的内容。

1.3.5 Current Directory 窗口

Current Directory 窗口主要显示的是当前在什么路径下进行工作，包括文件的保存等都是在当前路径下实现的。用户也可以选择 File | Set path 命令设置当前路径，如图 1.26 所示。

对该对话框中的各按钮说明如表 1.7 所示。

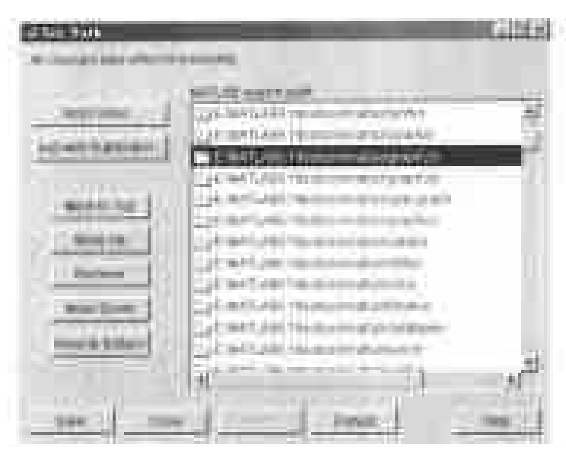


图 1.26 Set Path对话框

表 1.7 Set Path对话框中的按钮及其功能

按 钮	功 能
Add Folder	选择目录添加至 MATLAB search path(搜索路径)中
Add with Subfolders	将选中的目录路径的子目录也包含在搜索路径中
Move to Top	将搜索路径中选中的目录路径移至顶部
Move Up	将搜索路径中选中的目录路径向上移
Remove	清除搜索路径中选中的目录路径
Move Down	将搜索路径中选中的目录路径向下移
Move to Buttom	将搜索路径中选中的目录路径移至底部
Save	将设定好的搜索路径保存
Close	关闭 Set Path 窗口
Revert	撤销先前操作，恢复原状
Default	系统默认的搜索路径
Help	打开帮助窗口

1. MATLAB的路径搜索

MATLAB 采用路径搜索的方法来查找组织在文件系统中扩展名为.m 的 M 文件，常用的命令文件组织在 MATLAB 文件夹中，其他 M 文件组织在各种工具箱目录中。在 Command Window 窗口中输入一个字符串 matsear 时，将按以下的顺序开始搜索：

(1) 把 matsear 作为一个变量进行搜索，在当前工作空间中查找变量 matsear。

(2) 把 matsear 作为一个内置函数进行搜索，查找内置函数 matsear 并执行。

(3) 查找当前目录中的 matsear.m 文件。

(4) 查找当前搜索路径中的 M 文件 matsear.m。

此搜索的顺序只是一般情况下的顺序，而实际上的搜索规则还要考虑到私有函数、子函数和面向对象函数的范围限制，因此会更加复杂。但此简单的搜索顺序，对普通的 M 文件是非常准确的。

如果在搜索路径中存在同名函数，则仅可发现搜索路径中的第 1 个函数，而其他同名的函数不被执行。使用下面的一组命令可以对当前的搜索路径进行操作：

- Path：不加任何参数，显示当前搜索路径。
- path+[路径名]：设置当前的搜索路，以前的搜索路径无效。例如：

```
PATH E:\MATLAB 6\BIN
```

- addpath/sys/hhf 或 path(path,'')：向当前搜索路径中添加目录 sys/hhf，后者为函数形式。
- rmpath/sys/hhf：取消当前搜索路径中的目录 sys/hhf。
- pathtool：打开如图 1.26 所示的 Set Path 对话框设置搜索路径。

MATLAB 将默认的搜索路长保存在 pathdef.m 文件中，MATLAB 启动时将自动执行。此文件保存在\MATLAB\toolbox\local 目录下，用户可以使用文本编辑器直接对其进行编辑。如在 Command Window 窗口中输入：

```
Edit pathdef.m
```

2. Current Directory窗口菜单

Current Directory 窗口的菜单与 Command Window 窗口类似，只是在 View 菜单中增加了 Current Directory Filter 命令，如图 1.27 所示。该菜单用于设置在 Current Directory 窗口中显示的文件类型。



图 1.27 Current Directory Filter 子菜单

MATLAB 文件类型如表 1.8 所示。

表 1.8 MATLAB文件类型

文件类型	说　明	扩 展 名
M-files	M 文件	.m
MAT-files	数据文件	.mat
MEX-files	MEX 文件	.mex 或.dll
FIG-files	图形文件	.fig
P-files	P 文件	.p
Models	模型文件	.mdl

1.4 M 文件的编辑调试环境

MATLAB 的程序文件和脚本文件通常保存为扩展名为.m 的文件，本书称之为 M 文件。编辑 M 文件也可以用其他的文本编辑器，要启动 MATLAB 的 M 文件编辑器和调试器，可以在 Command Window 窗口中输入 Edit 命令，也可以选择 File | New/M-file 命令，或者单击图标按钮□。M 文件的编辑器和调试器如图 1.28 所示。

图 1.28 中的工具栏中有许多图标，下面只介绍其中一些图标的使用方法，其功能如表 1.9 所示。



图 1.28　M文件的编辑器和调试器

表 1.9　工具栏简介

图　标	功　能
	查找文本
	显示函数
	设置/清除断点
	清除所有断点
	继续调试过程
	遇到断点时单击此按钮可以转入被调用的函数或程序，可对其调试
	从设置断点函数中快速转出，继续调试
	运行调试的程序遇到断点后，单击此按钮继续执行
	退出程序调试

1.4.1　File 菜单



图 1.29　File菜单

File 菜单如图 1.29 所示，其中各命令的意义如下：

- New：新建 M 文件、图形、Simulink 模块。
- Open：打开 M 文件。
- Open Selection：打开选中的对象。
- Close/Save/Save as：关闭文件/保存文件/另存为。
- Save All：保存所有 M 文件。
- Source Control：该子菜单中的命令允许用户检查用户的源控制系统。
- Import Data：从外部文件导入数据。
- Save Workspace As：将当前工作空间另存为其他文件。
- Set Path：设置 MATLAB 的搜索路径。

- Preferences：设置 MATLAB 工作环境参数。
- Print：打印输出。
- Print Selection：打印所选中的文本或其他对象。
- Exit MATLAB：退出 MATLAB 系统。

1.4.2 Edit 菜单

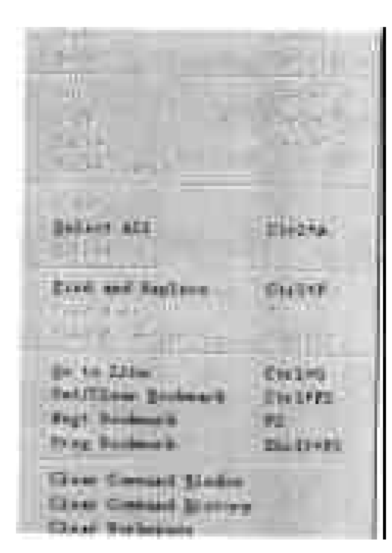


图 1.30 Edit菜单

Edit 菜单如图 1.30 所示，其中部分命令的意义如下：

- Undo：取消上一步操作。
- Redo：重新执行上一步操作。
- Cut：删除。
- Copy：复制。
- Paste：粘贴。
- Paste Special：选择性粘贴。
- Clear：清除对象。
- Select All：全部选取。
- Delete：删除对象。
- Find and Replace：查找并替代对象。
- Find Next：查找下一个对象。
- Find Selection：查找所选择的对象。
- Go to Line：跳转到指定的行。

1.4.3 Text 菜单

Text 菜单中的命令选项如图 1.31 所示。Text 菜单中的各命令的意义如下：



图 1.31 Text菜单

- Comment：注释程序行，选择该命令后，则鼠标指针所在的程序行无效。
- Uncomment：取消程序行注释，执行该命令后，则鼠标指针所在的程序行有效。
- Increase Indent：增加文本的缩进。
- Decrease Indent：减少文本的缩进。
- Balance Delimiters：平衡分界符。
- Smart Indent：智能缩进，即使用系统的设定对文本自动缩进处理。
- Evaluate Selection：计算所选部分表达式的值。

1.4.4 Debug 菜单



图1.32 Debug菜单

Debug 菜单如图 1.32 所示，Debug 菜单中各命令的意义如

下：

- Step：继续调试过程。
- Step In：运行调试的程序遇到断点后，选择此项可以转入被调用的函数或程序，并可对其调试。
- Step Out：快速调试，选择此命令，可从设置断点函数中快速转出，继续调试。
- Continue：运行调试的程序遇到断点后，选择此命令将继续执行。
- Go Until Cursor：执行到鼠标指针所在位置。
- Exit Debug Mode：退出程序调试模式。

1.4.5 Breakpoints 菜单

Breakpoints 菜单如图 1.33 所示，Breakpoints 菜单中各命令的意义如下：

- Set/Clear Breakpoints：设置/清除断点。
- Clear All Breakpoints：清除所有断点。
- Stop If Error：发生错误时停止执行程序。
- Stop If Warning：出现警告时停止执行程序。
- Stop If NaN Or Inf：遇到非数值或无穷大时停止执行程序。

图 1.33 Breakpoints菜单

1.4.6 Editor / Debugger 参数的设置

当用户使用 MATLAB Editor / Debugger 进行工作时，常常需要设置一些适合自己需要的环境，选择 File | Preferences 命令即可实现此目的。选择该命令后，系统将弹出如图 1.34 所示的 Preferences 对话框。

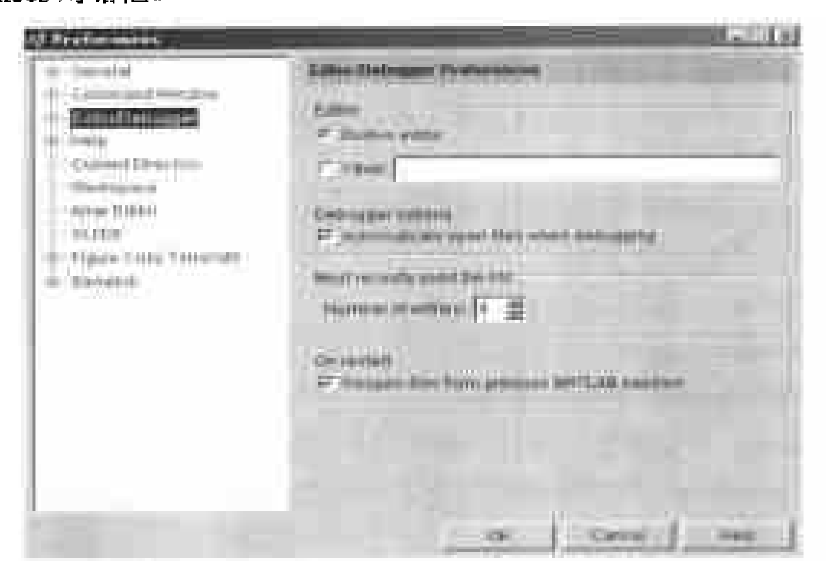


图 1.34 Preferences对话框

在该对话框中，可以设置各个窗口的参数，包括 Command Window 窗口、Editor/Debugger 窗口、Workspace 窗口等。例如，图 1.15 所示的 Preferences 对话框是用来设置

Command Window窗口的参数。这里只讲述Editor/Debugger的参数设置。

打开Preferences对话框后，如果要设置Editor/Debugger的参数，则可选择Editor/Debugger选项，进入如图1.34所示的环境。选择该选项，则弹出Editor/Debugger的参数设置选项。如图1.34所示，用户可以设置的基本参数如下：

- Editor选项组

  该选项组中的单选按钮用来设置用户将要使用的文本编辑器。

  - Built-in editor单选按钮表示使用MATLAB内置的编辑器。
  - Other单选按钮表示用户可以选择其他编辑器，此时可以在后面的文本框中输入编辑器的路径及应用程序名称。

- Debugger options选项组

  该选项组用来设置是否允许在调试时自动打开文件，选中Automatically open files when debugging复选框表示可在进行调试时自动打开文件。

- Most recently used files list选项组

  该选项组用来设置最近使用的文件列表数目，最大设置数目为9。

- On restart选项组

  该选项组用来设置重新运行系统时是否打开原来操作的文件。选中该选项组的复选框，表示下次启动MATLAB时，将打开上次退出MATLAB时正在编辑调试的文件。

1. Font & Colors选项

用户还可以设置Editor/Debugger的环境字体和颜色。在该对话框中选择Font & Colors选项，这时系统将弹出如图1.35所示的对话框，此时用户可以设置环境的字体和颜色。

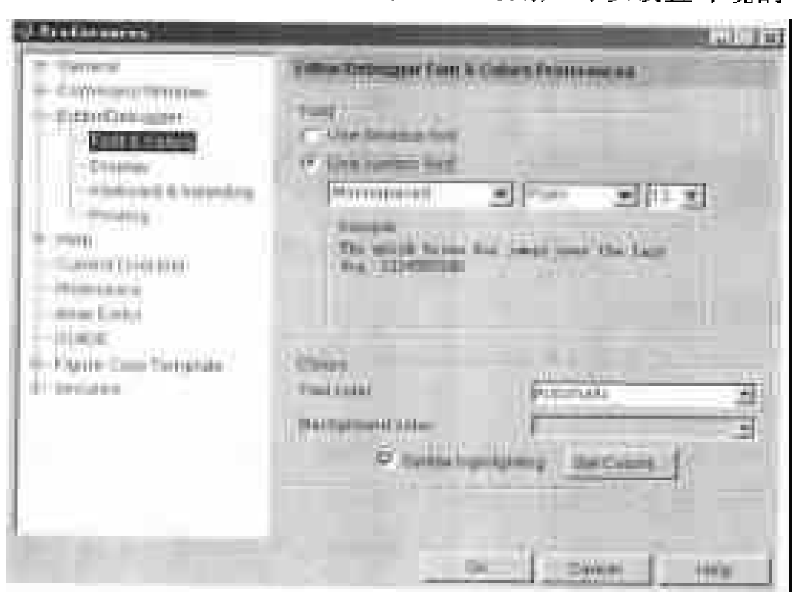


图1.35 选择Font & Colors选项

- 字体设置

  在如图1.35所示的对话框中，Font选项组用来设置字体。各选项的意义介绍如下：

  - 选择Use desktop font单选按钮，表示选用Windows桌面字体。选择该单选按

钮后，Font 选项组中的其他选项不可用。

- ◆ 选择 Use custom font 单选按钮，则可以设置用户自定义字体，此时用户可以分别在下面第 1 个下拉列表框选择需要设置的对象，然后在中间的下拉列表框中设置字体的类型，在第 3 个下拉列表框中选择字体的大小。

- 颜色设置

  在 Text color 下拉列表框中选择 Automatic 选项，系统会自动使用默认的颜色设置。如果选择了其他选项，则可以设置背景颜色。选中 Syntax highlighting 复选框，将突出显示语法。如果用户想自己设置突出显示的颜色，也可以单击 Set Colors 按钮，进入 General 标签设置其颜色。

2. Display选项

Editor/Debugger 的显示设置可以选择 Display 选项来设置，如图 1.36 所示，用户可以分别设置文件的打开方式和 Editor/Debugger 中的工具显示方式。

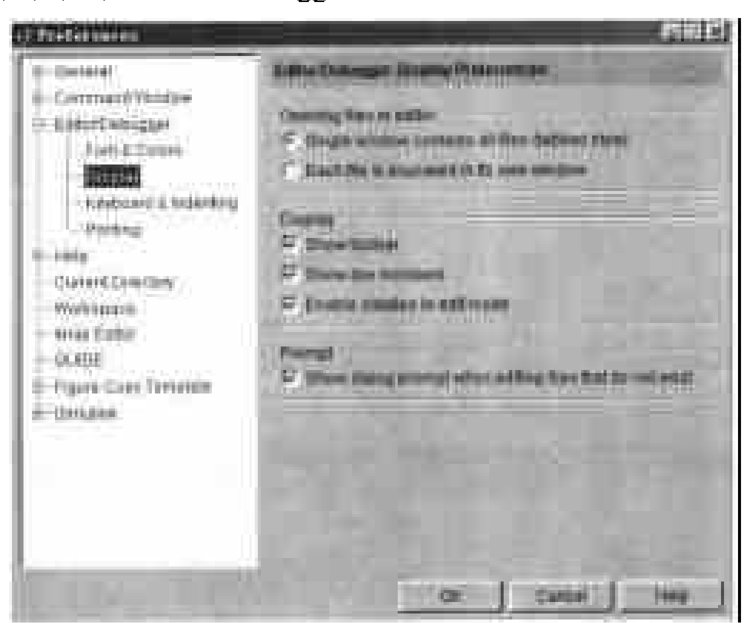


图 1.36 选择Display选项

- 设置文件的打开方式

  文件的打开方式可以在 Opening files in editor 选项组中进行设置，该选项组包括两个单选按钮。

  - ◆ Single window contains all files(tabbed style)单选按钮表示在一个窗口显示多个文件，各个文件以标签形式显示在左下角。
  - ◆ Each file is displayed in its own window 单选按钮表示每个文件在独立的窗口显示。

- 设置 Editor/Debugger 中的显示

  - ◆ 选择 Show toolbar 单选按钮，则在 Editor/Debugger 中显示工具栏。
  - ◆ 选择 Show line numbers 单选按钮，则在 Editor/Debugger 中显示文本的行数。
  - ◆ 选择 Enable datatips in edit mode 单选按钮，则显示数据提示。选择此单选按

钮后，显示鼠标指针所指的变量值。如图1.37所示，当鼠标指针停在变量上时，显示其内容。



图 1.37 显示数据提示

3. Keyboard & Indenting选项

编辑文本时，用户还可以在Preferences对话框中对键盘操作和字符缩进进行设置，单击Keyboard & Indenting选项即出现如图1.38所示的对话框。

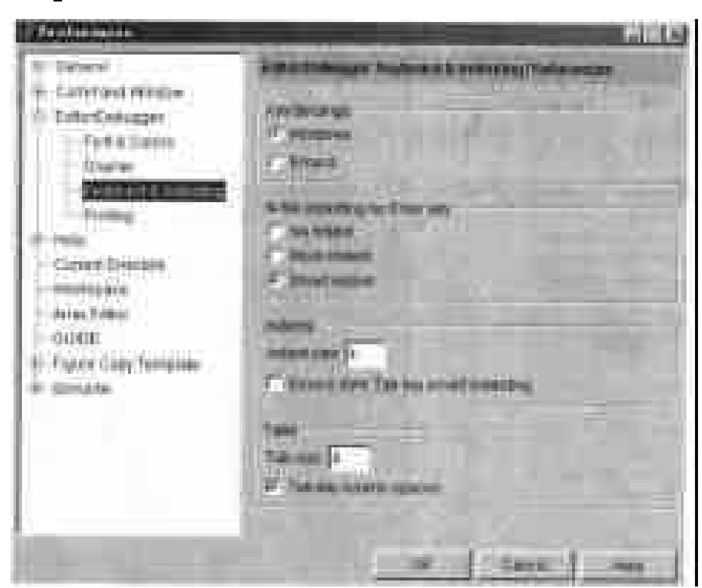


图 1.38 选择Keyboard & Indenting选项

- 键盘设定

  用户可以在Key bindings选项组中设置适合自己习惯的键盘定义。

  - 选择 Windows 单选按钮表示使用 Windows 约定的键盘快捷定义，如粘贴的快捷键为 Ctrl+V。
  - 选择 Emacs 单选按钮表示使用 Emacs 约定的键盘快捷定义，如粘贴的快捷键为 Ctrl+Y。

- M 文件缩进设置

  用户可以在 M-file indenting for Enter key 选项组为 M 文件设置不同的缩进方式，

分别如下：

- ◆ No indent：表示文本无缩进。
- ◆ Block indent：表示以块的形式缩进。
- ◆ Smart indent：表示采取智能缩进方式。

- ● 缩进参数设置

  用户还可以在 Indent 选项组设置适合于自己的缩进指数，比如缩进的字符数等。设置参数如下：

  - ◆ 在 Indent size 文本框输入同一标准的嵌套代码列数。
  - ◆ 选中 Emacs style Tab key smart indenting 复选框表示使用 Tab 键缩进当前行。

- ● 制表符设置

  用户可以在 Tab 选项组设置自己的制表符参数，设置选项如下：

  - ◆ Tab size 文本框中的值表示两个制表符 Tab 间的空格数。与 Auto indent size 中设置的数目不同，自动缩进只是插入空格，不插入制表符 Tab。
  - ◆ 选中 Tab key inserts spaces 复选框，则可选择插入一个 Tab 字符，或者与一个 Tab 字符或等效的空格数目。

4. Printing选项

另外用户可以选择 Printing 选项为 Editor/Debugger 的文本打印设置特定的参数，可以分别设置语法亮显(Syntax highlighting)的打印方式，是否打印文件头(即是否将文件路径和文件名等参数打印出来)，还可以专门设置打印字体，如图 1.39 所示。

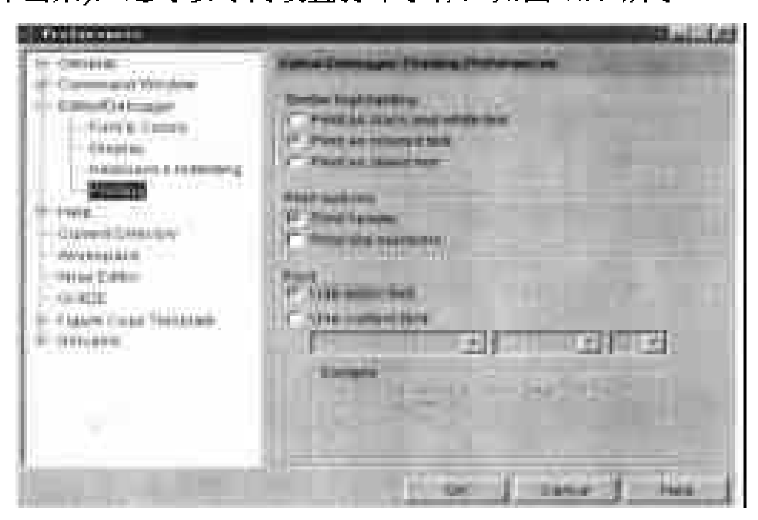


图 1.39 选择Printing选项

1.5 MATLAB 6 帮助

MATLAB 6 提供了强大的帮助系统，内容丰富，形式多样，MATLAB 6 的 Help 菜单所包括的命令如图 1.40 所示。

- Full Product Family Help 命令用于调用整个产品的帮助，选择该命令系统将显示如图 1.41 所示的帮助界面。
- 选择 MATLAB Help 命令，将打开 MATLAB 的帮助窗口，如图 1.42 所示，在 Help 窗口中的列表框中显示所有帮助的主题。单击某个帮助主题上，则自动跳转到相应的帮助内容。



图 1.40　帮助菜单

要查找某个函数的帮助信息，也可以在主题框中输入相应的函数名，或在主窗口中单击相应的主题。

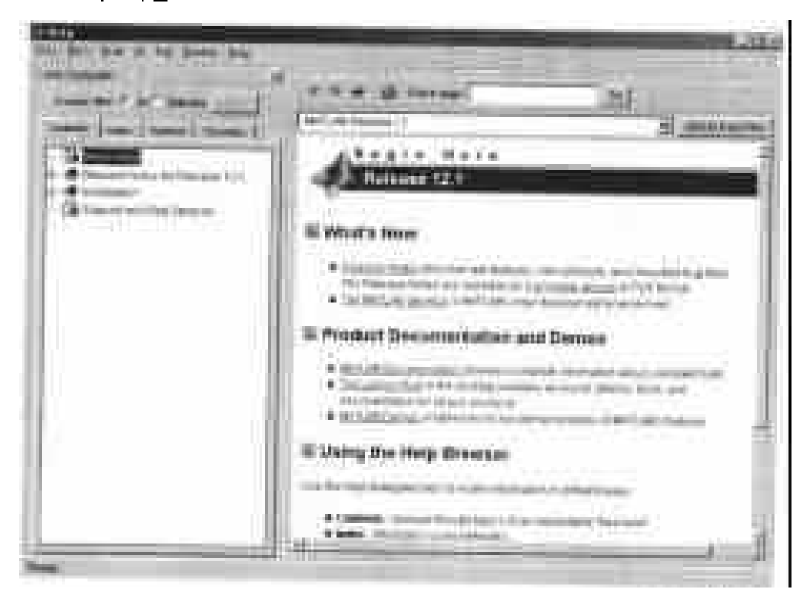


图 1.41　选择Full Product Family Help命令后的窗口

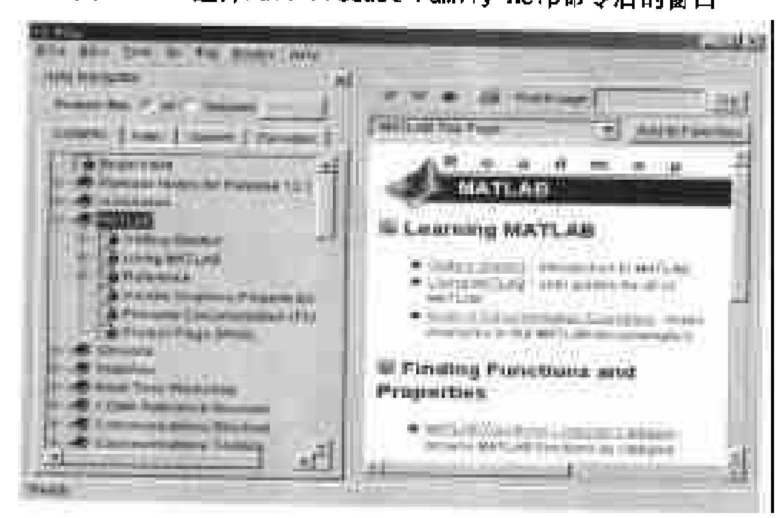


图 1.42　MATLAB的帮助窗口

- 选择 Using the Desktop 命令，将显示 MATLAB 界面的使用帮助。
- 选择 Using the Command Window 命令，将显示 Command Window 窗口使用帮助。
- 选择 Demos 命令，将显示如图 1.43 所示的对话框，提供各种例子及其图形演示。
- 选择 About MATLAB 命令，将显示 MATLAB 的启动界面及版本信息。

图 1.43 MATLAB Demo Window对话框

第2章　MATLAB 6基础

2.1　MATLAB表达式与变量

表达式和变量是使用 MATLAB 的基础，本节将介绍如何定义和使用表达式及变量，如何查看已经保存在内存中的变量，复数变量的定义和使用以及在 MATLAB 中有关永久变量的规定。另外，还将介绍相关的在屏幕中显示表达式和变量的格式设置。

2.1.1　MATLAB表达式

MATLAB 采用的是表达式语言，用户输入的语句由 MATLAB 系统解释运行。MATLAB 的语句是由表达式和变量组成，最常见的形式有以下两种：

- 表达式
- 变量=表达式

表达式由运算符、函数、变量名和数字组成。它在 MATLAB 中占有很重要的地位，几乎所有的运算都必须借助表达式来进行。

在第1种形式中，表达式运算后产生的结果如果是矩阵或其他的数值类型，MATLAB 系统将会自动赋给名为 ans 的变量，并显示在屏幕上。ans 是一个默认的变量名，它会在以后的类似操作中被自动覆盖掉。所以，对于重要的结果一定要记录下来，也就是要使用第2种形式。

在第2种形式中，等号右边的表达式表示计算后产生的结果，MATLAB 系统会将其赋给等号左边的变量，然后放入内存(工作空间)中，并显示在屏幕上。

在书写表达式时，运算符两侧允许有空格，以增加可读性。表达式的末尾可以加上“；”，也可以不加。有“；”时，MATLAB 系统不显示计算的结果，而是直接把数值赋给变量，如果没用变量就无法看到结果了。没有“；”时，MATLAB 系统将会在语句的下面显示运算的结果。

1.　**数字表达式**

MATLAB 的数值采用十进制表示，可以带小数点和负号；也可以使用科学计数法，用 e 表示位数，用以下表达式表示的数都是合法的。

```
1               0.1234          -65535
-1.234          1.2e34          3.456e-4
```

其中数值的相对精度为 $eps=2^{-52}$，数值的表示范围是 10^{-308}～10^{308}。

MATLAB 的数值也可以是复数，MATLAB 使用虚数单位 i 或 j 来定义复数，如：

```
i       j       3.14+1.732*i     1+j
```

2. MATLAB常用运算符

MATLAB 的常用运算符有+ (加法)、− (减法)、^ (幂)、*(乘法)、/ (右除)和\ (左除)。注意在矩阵运算中有左除和右除的区别，对于数字运算则没有区别。

接下来介绍几个创建无变量表达式的例子。

在 Command Window 窗口中输入：

```
1.2e34        %没有用变量赋值，MATLAB 系统自动赋给名为 ans 的变量。
ans=
1.2000e+034
(100*0.002)/4.0
ans =
    0.0500
3.14+1.732*i
ans =
   3.1400 + 1.7320i
```

2.1.2 MATLAB 的变量

和表达式紧密相关的是变量。除了 MATLAB 自定义的一些保留字以外，可以用一个字母打头，最后最多可接 19 个字母或数字定义一个变量。注意，在 MATLAB 中是区分大小写的。MATLAB 中不需要专门定义变量的类型，系统可以自动根据表达式的值或输入的值来确定变量的数据类型。因此，用户可以自由方便地使用变量。但是，如果使用和原来定义的变量一样的名字赋值，原变量将自动被覆盖，系统不会给出出错信息。使用变量时，要自觉地避免重复。

以下介绍几个创建有变量表达式的例子。

```
%表达式的赋值
a=(100*0.002)/4.0     %末尾没有“；”，按 Enter 键后直接显示结果。
a =
    0.0500
b=(2^10+2^9)/2^5
b =
    48
```

同样可以对复数变量进行赋值，如：

```
z1=1+i;
z2=1.i;
z3=z1*z2;
z4=z1/z2;
z1
z1 =
   1.0000 + 1.0000i
z2
z2 =
```

```
  0 + 1.0000i
z3
z3 =
    2
z4
z4 =
   0 + 1.0000i
```

定义变量时有些 MATLAB 变量不能用，其中有一部分就是 MATLAB 中的永久变量(有时也称为预定义的变量)。它们是在启动 MATLAB 时，系统自动定义的变量，驻留在内存中。它们不会被内存清除命令 clear 清除(因此称这些变量为永久变量)。用户可以为这些永久变量赋值，不过所赋的值可以用清除命令清除，从而恢复系统预定义的值(即所谓的预定义变量)。执行 who 命令，则看不到永久变量，表 2.1 列出了永久变量。

表 2.1 永久变量及其含义

变量名称	变量含义
Realmin 或 realmin	最小的浮点数，2^{-1022}
Realmax 或 realmax	最大的浮点数，2^{1023}
Eps	容差变量，定义为 1.0 到最近浮点的距离。PC 机上等于 2^{-52}
Pi	圆周率的近似值
Inf 或 inf	正无穷大，定义为(1/0)
NaN	非数(Not a Number)，产生于 0/0,∞/∞,0*∞等运算
i,j	虚数单位，定义为 $i=\sqrt{-1}$,$j=\sqrt{-1}$

下面是几个永久变量的例子：

```
a=sin(pi/2)    %sin()为 MATLAB 中的正弦函数。
a =
    1
%无穷大的使用。
r=1/0
Warning: Divide by zero.
r =
   Inf
```

注意： 在 MATLAB 中这样的操作不会引起程序执行中断，只是在给出警告信息的同时，用一个特殊的符号来表示。而且这个符号和其他的变量一样，可以在以后的运算中发挥作用。

```
1/r
ans =
    0
无穷大的作用。
r=0+realmin;
sin(r)/r
ans =
    1
realmin 的作用。
```

2.2 MATLAB 基本运算

MATLAB 具有高效的数值计算功能和强大的符号计算功能，本书只介绍它的基本运算。MATLAB 基本运算主要有数组和矩阵运算、关系和逻辑运算、字符和字符串运算以及符号运算等。

2.2.1 数值数组运算

事实上 MATLAB 是以数组(array)及矩阵(matrix)方式进行运算的，而这两者的基本运算性质不同，数组强调元素对元素的运算，而矩阵则采用线性代数的运算方式。

MATLAB 的数组类型有数值数组、字符串数组、元胞数组和构架数组，本书只讨论数值数组(数组元素是数值的数组)。

1. 一维数组的创建

创建数组的基本方法有 5 种，分别介绍如下：

- 直接创建，通过直接输入数组中每个元素的值来建立数组，方法是以左方括号开始，以空格或逗号为间隔输入元素值，最后以右方括号结尾。例如，在 Command Window 窗口中输入：

```
x=[1 2 3 4 5 6 7 8 9 10]
```

或者

```
x=[1,2,3,4,5,6,7,8,9,10]
```

按 Enter 键运行后，Command Window 窗口显示：

```
x =
1     2     3     4     5     6     7     8     9    10
```

- x=初值：终值，创建从初值开始到终值结束，增量为 1 的行向量 x，例如在 Command Window 窗口中输入：

```
x=10:20
```

按 Enter 键运行后，Command Window 窗口显示：

```
x =
  Columns 1 through 10
    10    11    12    13    14    15    16    17    18    19
  Column 11
20
```

- x=初值：增量：终值，创建从初值开始到终值结束，给定增量的行向量 x，例如在 Command Window 窗口中输入：

```
x=0:0.1:1
```

按 Enter 键运行后，Command Window 窗口显示：

```
x =
  Columns 1 through 3
                 0   0.10000000000000   0.20000000000000
  Columns 4 through 6
   0.30000000000000   0.40000000000000   0.50000000000000
  Columns 7 through 9
   0.60000000000000   0.70000000000000   0.80000000000000
  Columns 10 through 11
   0.90000000000000   1.00000000000000
```

- x=linspace(初值，终值，n)，创建从初值开始到终值结束，有 n 个元素的行向量 x，例如在命令窗口中输入：

```
x=linspace(0,pi,11)
```

或者

```
x=linspace(0,1,11)*pi
```

按 Enter 键运行后，命令窗口显示：

```
x =
  Columns 1 through 3
                 0   0.31415926535898   0.62831853071796
  Columns 4 through 6
   0.94247779607694   1.25663706143592   1.57079632679490
  Columns 7 through 9
   1.88495559215388   2.19911485751286   2.51327412287183
  Columns 10 through 11
   2.82743338823081   3.14159265358979
```

- x=logspace(初值，终值，n)，创建从初值开始到终值结束，有 n 个元素对数分隔量]x，例如在命令窗口中输入：

```
x=logspace(0,1,11)
```

按 Enter 键运行后，命令窗口显示：

```
x =
  Columns 1 through 7
   1.0000    1.2589    1.5849    1.9953    2.5119    3.1623    3.9811
  Columns 8 through 11
   5.0119    6.3096    7.9433   10.0000
```

此时的数据输出格式为 short 型。

2. 一维子数组寻访和赋值

MATLAB 用括号来表示下标，访问数组中的元素同样是利用元素的下标。x(n) 表示数组中的第 n 个元素。利用冒号，MATLAB 还可以同时对多个元素进行访问，如下例所示：

```
rand('state',0)        %把均匀分布伪随机发生器置为 0 状态。
```

```
x=rand(1,5)           %产生 1×5 的均布随机数组。
x =
0.9501    0.2311    0.6068    0.4860    0.8913
x(3)                %寻访数组 x 的第 3 个元素。
ans =
    0.6068
x([1 2 5])  %寻访数组 x 的第 1、2、5 个元素组成的子数组。
ans =
    0.9501    0.2311    0.8913
x(1:3)              %寻访前 3 个元素组成的子数组。
ans =
   0.9501    0.2311    0.6068
x(3:end)            %寻访除前两个元素外的全部其他元素。end 是最后一个元素的下标。
ans =
   0.6068    0.4860    0.8913
x(3:-1:1)           %由前 3 个元素倒排构成的子数组。
ans =
   0.6068    0.2311    0.9501
x(find(x>0.5)) %由大于 0.5 的元素构成的子数组。
ans =
0.9501    0.6068    0.8913
x([1 2 3 4 4 3 2 1]) %对元素可以重复寻访，使所得数组长度允许大于原数组。
ans =
  Columns 1 through 7
    0.9501    0.2311    0.6068    0.4860    0.4860    0.6068    0.2311
  Column 8
    0.9501
```

如果数组中只有部分元素需要改动，则可通过子数组赋值来实现，如下例所示：

```
x(3)=0              %把上例中的第 3 个元素重新赋值为 0。
x =
    0.9501    0.2311         0    0.4860    0.8913
x([1 4])=[1 1]  %把当前 x 数组的第 1、4 个元素都赋值为 1。
x =
1.0000    0.2311         0    1.0000    0.8913
```

3. 二维数组

二维数组可以看成一维数组的扩展，性质与一维数组类似。如下面的例子所示：

```
a=[1,2,3;3,4,5]       %生成一个 2×3 的二维数组。
a =                   %运行结果。
    1     2     3
    3     4     5
rand('state',0)       %把均匀分布伪随机发生器置为 0 状态。
x=rand(3,5)           %产生 1×5 的均布随机数组。
x =
    0.9501    0.4860    0.4565    0.4447    0.9218
    0.2311    0.8913    0.0185    0.6154    0.7382
    0.6068    0.7621    0.8214    0.7919    0.1763
x(3,1)              %寻访数组 x 的第 3 行，第一列元素。
```

```
ans =
        0.6068
```

4. 两种特殊数组

元素全为1的数组称为1数组，元素全为0的数组称为零数组。

MATLAB用函数ones可以创建1数组，其一般用法为：

- ones(n)　创建一个n×n的1数组。
- ones(r,c)　创建一个r×c的1数组。

例如：

```
ones(3)    %创建一个3×3的1数组。
ans =
     1     1     1
     1     1     1
     1     1     1
ones(3,6)  %创建一个3×6的1数组。
ans =
     1     1     1     1     1     1
     1     1     1     1     1     1
     1     1     1     1     1     1
```

MATLAB用函数zeros可以创建零数组，其用法与ones相同。例如：

```
zeros(3)    %创建一个3×3的0数组。
ans =
     0     0     0
     0     0     0
     0     0     0
zeros(3,6)  %创建一个3×6的0数组。
ans =
     0     0     0     0     0     0
     0     0     0     0     0     0
     0     0     0     0     0     0
```

5. 数组运算

数组运算是元素对元素的运算，与矩阵运算不同。

数组运算及功能如下，其中除了加减符号外，其余的数组运算符号均须多加“.”符号。

+	加	./	左除
−	减	.^	乘幂
.*	乘	.'	转置

举例如下：

```
a=1:5;a-2               % 从数组a减2。
ans =
-1 0 1 2 3
2*a-1       %以2乘数组a再减1。
ans =
    1 3 5 7 9
```

```
b=1:2:9;a+b     % 数组a加b。
ans =
2 5 8 11 14
a.*b     % 数组a及b中的元素与元素相乘。
ans =
1 6 15 28 45
a./b      % 数组a及b中的元素与元素相除。
ans =
1.0000 0.66667 0.6000 0.5714 0.5556
a.^2      % 数组中的各个元素作二次方。
ans =
1 4 9 16 25
2.^a % 以2为底，以数组中的各个元素为次方。
ans =
2 4 8 16 32
b.^a % 以数组b中的各个元素为底，以数组a中的各个元素为次方。
ans =
1 9 125 2401 59049
b=a' % 数组b是数组a的转置结果。
b =
     1
     2
     3
     4
     5
a=ones(2,3);
b=zeros(2,3);
a+b+a
ans =
     2     2     2
     2     2     2
a.*b    % 数组a及b中的元素与元素相乘。
ans =
     0     0     0
     0     0     0
```

6. 数组(矩阵)操作

对数组或矩阵的基本操作有插入、重新排列、提取、按列拉长、置空(去掉某行或某列)、置零、用单个下标操作一个矩阵、用逻辑数组操作一个矩阵、按指定条件求子数组、求数组的规模等，下面一一举例说明(对数组和矩阵不加区别)。

```
x=4:6
x =
     4     5     6
```

- 插入

  通过对x进行插入运算创建矩阵A。

```
A=[x-3;x;x+3]
A =
```

```
      1       2       3
      4       5       6
      7       8       9
```

- 重新排列

  以逆序重排 A 的各行形成矩阵 B。

```
B=A(3:-1:1,1:3)
B =
      7       8       9
      4       5       6
      1       2       3
```

- 提取

  提取的 A 前两行的后两列形成矩阵 C。

```
C=A(1:2,2:3)
C =
      2       3
      5       6
```

- 按列拉长

  对 C 按列拉长形成矩阵 D。

```
D=C(:)
D =
      2
      5
      3
      6
```

- 置空(去掉某行或某列)

  删除 B 的第 2 列。

```
B(:,2)=[]
B =
      7       9
      4       6
      1       3
```

- 置零

  将矩阵 B 的第 2 行第 2 列的元素置为 1。

```
B(2,2)=0
B =
      7       9
      4       0
      1       3
```

- 用单个下标操作一个矩阵

  MATLAB 对矩阵中的元素赋予一个序号，序号值按列从第 1 列第 1 行到第 1 列第 2 行到第 2 列第 1 行到第 2 列第 2 行，直至最后一列最后一行的顺序计数。

```
%获得矩阵 B 的第 3 个元素。
B(3)
ans =
     1
%获得矩阵 B 的第 5 个元素。
B(5)
ans =
     0
%获得矩阵 B 的第 2 至第 4 个元素。
B(2:4)
ans =
     4     1     9
```

- 用逻辑数组操作一个矩阵

```
x= -4:4
x =
    -4    -3    -2    -1     0     1     2     3     4
```

%判断数组 x 中元素的绝对值大于 3 的情况，绝对值大于 3 的元素置为 1，否则为 0。

```
abs(x)>3
ans =
     1     0     0     0     0     0     0     0     1
```

对矩阵可进行同样操作：

```
A=[1,2,3;4,5,6;7,8,9]
A =
     1     2     3
     4     5     6
     7     8     9
abs(A)>3
ans =
     0     0     0
     1     1     1
     1     1     1
```

- 按指定条件求子数组

首先由条件表达式生成一个逻辑数组，然后使用系统提供的函数 find 即可。函数 find 在已知数组中按逻辑数组的元素值查找所需的元素下标。

接上例，

```
a=abs(x)>2
a =
     1     1     0     0     0     0     0     1     1
b=find(a)    %返回数组中值不为零的那些元素的下标。
b =
     1     2     8     9
y=x(b)       %提取 x 中下标数组 b 对应的元素。
y =
```

```
    -4    -3     3     4
```

函数 find()也可用于矩阵。

```
A=[1:3;4:6;7:9]
A =
     1     2     3
     4     5     6
     7     8     9
B=A>4
B =
     0     0     0
     0     1     1
     1     1     1
[r,s]=find(B)       %返回矩阵 B 中值不为零的元素的行列下标。
r =
     3
     2
     3
     2
     3
s =
     1
     2
     2
     3
     3
```

- 求数组的规模

  数组的规模就是这个数组的大小，即这个数组的行数和列数。MATLAB 提供了两个函数 size 和 length 来获得已知数组的规模。

  函数 size 和 length 的使用方法如下：

  - s=size(A)

    返回一个二元素行向量 s，s 的第 1 个元素为 A 的行数，第 2 个元素为 A 的列数。

    ```
    A=[3:6;1:4]
    A =
         3     4     5     6
         1     2     3     4
    s= size(A)
    s =
           2     4
    ```

  - [r,c]=size(A)

    返回两个变量 r 和 c 分别为 A 的行数和列数。

    ```
    [r,c]=size(A)
    r =
         2
    c =
         4
    ```

- r=size(A,1)
  只返回 A 的行数。

```
r=size(A,1)
r =
     2
```

- c=size(A,2)
  只返回 A 的列数。

```
c=size(A,2)
c =
     4
```

- n=length(A)
  返回行数和列数中较大的一个，即 max(size(A))。

```
n=length(A)
n =
     4
```

- 数组(矩阵)操作函数
  MATLAB 提供了几种函数来运行数组操作功能，如表 2.2 所示。

表 2.2 数组操作函数

函数名	功能
diag(A)	提取矩阵 A 的对角元素，并返回给列向量
diag(v)	以向量 v 作对角元素来创建对角矩阵
flipud(A)	将矩阵上下翻转
fliplr(A)	将矩阵左右翻转
rot90(A)	矩阵逆时针翻转 90°
reshape(A,m,n)	返回一个 m×n 矩阵，其元素是以列方式从 A 中获得，A 必须包含 m×n 个元素
tril(A)	提取矩阵 A 的下三角矩阵
triu(A)	提取矩阵 A 的上三角矩阵

举几个例子说明如下：

```
A=[1,8,7;2,9,6;3,4,5]
A =
     1     8     7
     2     9     6
     3     4     5
v=diag(A)
v =
     1
     9
     5
```

```
diag(v)
ans =
     1     0     0
     0     9     0
     0     0     5
flipud(A)
ans =
     3     4     5
     2     9     6
     1     8     7
fliplr(A)
ans =
     7     8     1
     6     9     2
     5     4     3
rot90(A)
ans =
     7     6     5
     8     9     4
     1     2     3
tril(A)
ans =
     1     0     0
     2     9     0
     3     4     5
triu(A)
ans =
     1     8     7
     0     9     6
     0     0     5
```

7. **多维数组**

多维数组是指三维以至更多维数的数组。关于多维数组这里只举一个例子，有关内容请参看其他参考书。

```
B=[1,2;3,4];
B(:,:,2)=[5,6;7,8];
B
B(:,:,1) =
      1     2
      3     4
B(:,:,2) =
      5     6
      7     8
```

2.2.2 矩阵运算

矩阵的范围比数组窄的多，通常可以认为矩阵是一维或二维的数值数组(实数组或复数组)，二者在 MATLAB 的基本运算性质也不同，数组强调元素对元素的运算，而矩阵则采

用线性代数的运算方式。

矩阵的创建和访问与一维或二维的数值数组相同，此处不再赘述。

1. 常用的特殊矩阵

MATLAB 中用于产生常用特殊矩阵的函数如表 2.3 所示。

表 2.3 常用特殊矩阵的函数

矩阵函数	说　明
zeros(m,n)	零矩阵
ones(m,n)	1 矩阵
eye(m)	单位矩阵
randn(m,n)	正态分布的随机矩阵
company(A)	矩阵 A 的伴随矩阵
gallery	测试矩阵
hankel(m,n)	n 维 Hankel 矩阵
invhilb(n)	n 维 Hilbert 逆矩阵
Magic(n)	n 维 Magic 方阵
toeplitz(m,n)	Toeplitz 矩阵
wilkinson(n)	n 维 Wilkinson 特征值测试矩阵
hadamard(n)	n 维 Hadamard 矩阵
Hilb(n)	n 维 Hilbert 矩阵
kron(A,B)	Kronecker 张量积
pascal(n)	n 维 Pascal 矩阵
vander(A)	由矩阵 A 产生 Vandermonde 矩阵

用户可以在 Command Window 窗口中试验一下，例如：

```
eye(3)
ans =
     1     0     0
     0     1     0
     0     0     1
randn(3,2)
ans =
   -0.4326    0.2877
   -1.6656   -1.1465
    0.1253    1.1909
magic(3)
ans =
     8     1     6
     3     5     7
     4     9     2
hilb(3)
ans =
```

```
  1.0000    0.5000    0.3333
  0.5000    0.3333    0.2500
  0.3333    0.2500    0.2000
```

2. 矩阵运算

矩阵和数组运算是不同的，数组运算是元素对元素的运算，而矩阵则采用线性代数的运算方式，两种运算符号也不同，它们的区别如表 2.4 所示。

表 2.4　数组运算符号和矩阵的基本运算符号

数组运算符号	矩阵运算符号	功　能
+	+	相加
.*	*	相乘
./	/	左除
.\	\	反除
.^	^	乘幂

对比矩阵和数组的基本运算，二者加减运算都是关于对应元素的加减，但乘、除、乘幂运算二者有所不同。例如：

```
A=eye(2)
A =
     1     0
     0     1
B=[1,2;3,4]
B =
     1     2
     3     4
A+B
ans =
     2     2
     3     5
A.*B
ans =
     1     0
     0     4
A*B
ans =
     1     2
     3     4
A./B
ans =
    1.0000         0
         0    0.2500
A/B
ans =
   -2.0000    1.0000
    1.5000   -0.5000
```

```
B.^2
ans =
     1     4
     9    16
A^2
ans =
     1     0
     0     1
B.^A
ans =
     1     1
     1     4
```

矩阵的基本运算还包括指数运算、对数运算和开方运算。

2.2.3 数组函数和矩阵函数

MATLAB 提供的函数分为两类：一类是按数组运算法设计的，称之为数组函数，用 f(·)表示；另一类是按矩阵运算法则设计的，称为矩阵函数，用 funm(·)表示。

1. 基本数组函数

数组函数是按以下规则设计的：

对于 A=(aij)m×n，定义 f(A)=(f(aij)) m×n 。

满足这个定义的基本函数命令如表 2.5 和表 2.6 所示。

表 2.5 基本数组函数表

函数名称	功　能	函数名称	功　能
sin	正弦	acosh	反双曲余弦
cos	余弦	atanh	反双曲正切
tan	正切	acoth	反双曲余切
cot	余切	asech	反双曲正割
sec	余割	acsch	反双曲余割
csc	正割	fix	朝零方向取整
asin	反正弦	ceil	朝正无穷大方向取整
acos	反余弦	floor	朝负无穷大方向取整
atan	反正切	round	四舍五入到整数
atan2	四象限反正切	rem	除后取余数
acot	反余切	sign	符号函数
asec	反正割	abs	绝对值
acsc	反余割	angle	复数相角
sinh	双曲正弦	imag	复数虚部
cosh	双曲余弦	real	复数实部

续表

函数名称	功　能	函数名称	功　能
tanh	双曲正切	conj	复数共轭
coth	双曲余切	log10	常用对数
sech	双曲余割	log	自然对数
csch	双曲正割	exp	指数
asinh	反双曲正弦	sprt	平方根

表 2.6　特殊数组函数表

函数名称	功　能	函数名称	功　能
bessel	第一、第二类函数	erf	误差函数
beta	函数	eerfinv	逆误差函数
gamma	函数	ellipk	第一、第二类全椭圆积分
rat	有理近似	ellipj	椭圆函数

2. 基本矩阵函数

MATLAB 提供的基本矩阵函数指令如表 2.7 所示。

表 2.7　基本矩阵函数表

函数名称	功　能
cond(A)	矩阵 A 的条件数
det(A)	矩阵 A 的行列式值
dot(A,B)	矩阵 A、B 的点积
eig(A)	矩阵 A 的特征值和特征向量
norm(A,1)	矩阵 A 的 1-范数
norm(A)或 norm(A,2)	矩阵 A 的 2-范数
norm(A,inf)	矩阵 A 的无穷大范数
norm(A,'fro')	矩阵 A 的 F-范数
rank(A)	矩阵 A 的秩
rcond(A)	矩阵 A 的倒条件数
svd(A)	矩阵 A 的奇异值分解
trace(A)	矩阵 A 的迹
expm(A)	矩阵的指数
expm1(A)	用法求矩阵的指数
expm2 (A)	用级数求矩阵的指数，精度差，对任何矩阵都适用
expm3(A)	用特征值和特征向量求矩阵的指数，仅当独立特征向量个数等于矩阵秩时适用

续表

函数名称	功　能
logm(A)	求矩阵 A 的对数
sqrtm(A)	求矩阵 A 的平方根
funm(A,'fun')	一般的方阵函数

下面只对表中的一些较典型的函数作详细说明。

- MATLAB 中求矩阵 A 的范数的函数用法如下：

  当 n=norm(A)时，计算矩阵 A 的最大奇异值，相当于 n=max(svd(A))；当 n=norm(A,p)时，根据参数 p 值的不同，求不同的范数值。

  - 当 p 为 1：计算矩 A 的 1－范数，或列和的最大值相当于 n=max(sum (abs(A)))。
  - 当 p 为 2：计算矩阵 A 的最大奇异值，与 n=norm(A)相同。
  - 当 p 为 inf：计算矩阵 A 的无穷范数，或行和的最大值相当于 n=max(sum (abs(A')))；
  - 当 p 为'fro'：计算矩阵 A 的 F－范数(Frobenius－范数)，相当于 n=sqrt(sum(diag (A'*A)))。

  MATLAB 中求向量 X 的范数的函数用法如下：

  当 n=norm(x,p)时，对任意 1≤P≤∞，返回值 sum(abs(x).^p)^(1/p)；当 n=norm(x)时，返回值 n=norm(x,2)；当 n=norm(x,inf)时，返回值 max(abs(x))；n=norm(x,-inf)时，返回值 min(abs(x))。

  举例如下：

```
A=[1.5,0,0,0,0;0,2.5,0,3.5,0;0,0,4.5,0,0;0,0.5,0,5.5,0;0,0,0,0,1] %定义矩阵。
A =
     1.5000       0          0          0        0
        0      2.5000        0       3.5000      0
        0         0       4.5000        0        0
        0      0.5000        0       5.5000      0
        0         0          0          0     1.0000
n=norm(A)
n =
    6.7720
ninf=norm(A,inf)
ninf=
     6
n_inf=norm(A,-inf)
n_inf =
      6
n2=norm(A,2)
n2 =
     6.7720
n1=norm(A,1)
n1 =
     9
```

- 矩阵的特征值与特征向量运算
  - [V，D]=eig(A) D为对角阵，其对角线上为A的特征值，每个特征值对应矩阵V的列为该特征值的特征向量，该矩阵为一满秩矩阵，AV＝VD；且每个特征向量各元素的平方和(即2－范数)均为1。
  - d=eig(A) 返回矩阵A的特征值向量。

  如果特征向量矩阵V的条件数特别大，可以被认为是病态矩阵，此时再用V进行操作和运算是不合适的，实际中要对原始矩阵作一特殊的相似变换，即所谓的均衡变换，引入一个矩阵F，使变换后的矩阵B满足 $FBF^{-1}=A$；在MATLAB中用以下方法实现：
  - [F,B]=balance(A) 返回对角矩阵F和均衡矩阵B，如果A为对称矩阵，则A=B，F为单位矩阵。
  - B=balance(A) 仅返回均衡矩阵B。

MATLAB计算特征值函数eig(A)自动进行均衡，可用[V,D]=eig(A,'nobalance')来关闭均衡。

除上述的特征值外，还有广义特征值，MATLAB中的广义特征值求法为：
- [V,D]= eig(A,B) 返回一个满秩特征向量矩阵V及一个对角特征值矩阵D，满足A*V=B*V*D。
- D＝eig (A,B) 直接返回广义特征值向量。

举例如下：

```
A=[1.5,0,0,0,0;0,2.5,0,3.5,0;0,0,4.5,0,0;0,0.5,0,5.5,0;0,0,0,0,…]
%定义矩阵。
A =
    1.5000      0          0         0         0
       0     2.5000        0      3.5000       0
       0        0       4.5000       0         0
       0     0.5000        0      5.5000       0
       0        0          0         0      1.0000
V,D]=eig(A)
V =
       0        0       1.0000       0         0
    0.7071   0.9899        0         0         0
       0        0          0      1.0000       0
    0.7071  -0.1414        0         0         0
       0        0          0         0      1.0000
D =
    6.0000      0          0         0         0
       0     2.0000        0         0         0
       0        0       1.5000       0         0
       0        0          0      4.5000       0
       0        0          0         0      1.0000
d=eig(A)
d =
   6.0000
   2.0000
   1.5000
```

```
    4.5000
    1.0000
A*V
ans =
        0         0        1.5000        0         0
   4.2426    1.9799          0           0         0
        0         0          0        4.5000       0
   4.2426   -0.2828          0           0         0
        0         0          0           0       1.0000
V*D
ans =
        0         0        1.5000        0         0
   4.2426    1.9799          0           0         0
        0         0          0        4.5000       0
   4.2426   -.02828          0           0         0
        0         0          0           0       1.0000
[F,B]= balance(A)
F =
        0         0        1.0000        0         0
        0      1.0000        0           0         0
        0         0          0        1.0000       0
   0.5000         0          0           0         0
        0         0          0           0       1.0000
B =
   5.7000    1.0000          0           0         0
   1.7500    2.5000          0           0         0
        0         0        1.5000        0         0
        0         0          0        4.5000       0
        0         0          0           0       1.0000
[V,D]= eig(A, B)
V =
  -1.0000    0.2773       -0.2183        0         0
  -0.3112    1.0000        1.0000        0         0
        0         0          0        1.0000       0
   0.0359   -0.2583        0.2723        0         0
        0         0          0           0       1.0000
D =
   0.2581         0          0           0         0
        0    00.7921         0           0         0
        0         0        1.6303        0         0
        0         0          0        3.0000       0
        0         0          0           0       1.0000
A*V
ans =
  -1.5000   -0.4159       -0.3275        0         0
  -0.6526    1.5959        3.4529        0         0
        0         C          0        4.5000       0
   0.0417   -0.9208          0                   1.0000
        0         0          0           0       1.0000
B*V*D
ans =
```

```
    -1.5000   0.4159    -0.3275      0      0
    -0.6526   1.5959     3.4529      0      0
          0        0          0   4.5000    0
          0        0          0   4.5000    0
     0.0417  -0.9208     1.9974      0      0
```

- 矩阵求逆及广义逆

MATLAB 中求逆函数如下：

C=inv(A) 返回方阵 A 的逆，如果 A 为奇异或接近奇异矩阵，则产生出错信息。如果需要求奇异矩阵的一种“逆”阵，这种“逆”就称为矩阵的广义逆，又叫伪逆，在 MATLAB 中求矩阵广义逆的函数为：

```
N=pinv(A)   %返回矩阵 A 的广义逆，判 0 误差限为机器精度 eps。
N=pinv(A, tol)  %返回矩阵 A 的广义逆，用 tol 作为判 0 误差限。
```

此时，AN 和 NA 不一定是单位矩阵，它满足 ANA=A 和 NAN=N 两个条件。

如果 A 是一个 n×m 阶长方形矩阵，则用函数 pinv()也可求出矩阵 A 的广义逆矩阵，在数字图像处理中，常遇到求解问题：g=Af，g 与 f 代表向量，A 表示矩阵。如果 A 为方阵，且存在逆矩阵，则其解唯一表示为 $f=A^{-1}g$；若 A 为 m×n 阶长方形矩阵，或为行列式值为 0 的方阵，而不能直接求逆，则可根据广义逆矩阵理论，求得 f 的最佳估计值。

举例如下：

```
A=[1.5,0,0,0,0;0,2.5,0,3.5,0;0,0,4.5,0,0;0,0.5,0,5.5,0;0,0,0,0,1]
%定义矩阵。

A =
    1.5000        0         0         0        0
         0   2.5000         C    3.5000        0
         0        0    4.5000         0        0
         0   0.5000         0    5.500C        0
         0        0         0         0   1.0000
X=inv(A)
X =
    0.6667        0         0         0        0
         0   0.4583         0   -0.2917        0
         0        0    0.2222         0        0
         0  -0.0417         0    0.2083        0
         0        0         0         0   1.0000
N=pinv(A)
N =
    0.6667        0         0         0        0
         C   0.4583         0   -0.2917        0
         0        0    0.2222         0        0
         0  -0.0417         0    002083        0
         0        0         0         0   1.0000
A*V
ans =
   -1.5000  -0.4159   -0.3275         0        0
   -0.6526   1.5959    3.4529         0        0
```

```
        0         0         0       4.5000    0
        0         0         0       4.5000    0
     0.0417   -0.9208    1.9974       0       0
        0         0         0         0     1.0000
X*A
ans =
     1.0000       0         0         0       0
        0      1.0000       0         0       0
        0         0      1.0000       0       0
        0     -0.0000       0      1.0000     0
        0         0         0         0     1.0000
B=[1,2,3,4;4,5,6,7;2,4,6,8;1,3,5,7]
B =
     1    2    3    4
     4    5    6    7
     2    4    6    8
     1    3    5    7
Y= inv(B)
Warning: Matrix is singular to working precision.
Y =
   Inf   Inf   Inf   Inf
   Inf   Inf   Inf   Inf
   Inf   Inf   Inf   Inf
   Inf   Inf   Inf   Inf
N=pinv(B)
N =
   -0.0404    0.2717   -0.0808   -0.1848
   -0.0152    0.1394   -0.0303   -0.0818
    0.0101    0.0071    0.0202    0.0212
    0.0354   -0.1253    0.0707    0.1242
B*N*B
ans =
    1.0000    2.0000    3.0000    4.0000
    4.0000    5.0000    6.0000    7.0000
    2.0000    4.0000    6.0000    8.0000
    1.0000    3.0000    5.0000    7.0000
N*B*N
ans =
   -0.0404    0.2717   -0.0808   -0.1848
   -0.0152    0.1394   -0.0303   -0.0818
    0.0101    0.0071    0.0202    0.0212
    0.0354   -0.1253    0.0707    0.1242
```

- 矩阵的特征多项式、特征方程与特征根

  MATLAB 中提供了求矩阵特征多项式系数的函数，其调用格式如下：

  c=poly(A)　A 为 n 阶方阵，返回一个由 n+1 个元素组成的行向量 c，其各个分量为矩阵 A 的降幂排列的特征多项式系数。

  MATLAB 中求矩阵特征根的函数调用方法如下：

  - r=roots(c)　c 为特征多项式的系数向量，而所求得的列向量 r 为特征方程的解，即原矩阵的特征根。在已知矩阵的特征根向量 r 后，也可求得原矩阵的

特征多项式系数。

- c=poly(r) r 为矩阵的特征根向量，返回一个行向量 c，其各个分量为原矩阵 A 的降幂排列的特征多项式系数。

对于多项式来说，有时要求它的值，可以通过如下 MATLAB 函数调用得出：

v=polyval(c,x) 返回降幂排列多项式系数为向量 c 的多项式，在 x 处的值，x 也可以为向量或矩阵，此时，则逐个计算各元素的值。

对于矩阵多项式的值，还可用如下函数：

B=polyval(c,X) 降幂排列多项式系数向量 C，X 为方阵，其计算结果为：

$B=c1X^{n}+c2X^{n-1}+\cdots+cnX+cn.1I$

举例如下：

```
A=[1.5,0,0,0,0;0,2.5,0,3.5,0;0,0,4.5,0,0;0,0.5,0,5.5,0;0,0,0,0,1]
%定义矩阵
A =
    1.5000       0          0          0       0
         0  2.5000          0     3.5000       0
         0       0     4.5000          0       0
         0  0.5000          0     5.5000       0
         0       0          0          0  1.0000
c=poly(A)
c =
   1.0000  -15.0000   80.7500 -192.7500  207.0000  -81.0000
r=roots(c)
r =
   6.0000
   4.5000
   2.0000
   1.5000
   1.0000
c=poly(r)
c =
   1.0000  -15.0000   80.7500 -192.7500  207.0000  -81.0000
X=[1 1 1; 2 2 2; 3 3 3]
X =
    1     1     1
    2     2     2
    3     3     3
a=[1 1 1]
a =
    1     1     1
f=polyval(a,X)
f =
    3     3     3
    7     7     7
   13    13    13
```

3. 矩阵分解函数

矩阵的分解在代数中占有重要地位，是矩阵和数据分析的基础。MATLAB 系统提供大量的矩阵分解函数，如表 2.8 所示。

表 2.8 矩阵分解函数

函数名称	功 能
cdf2rdf(V,D)	复数对角形转换成实数块对角形
chol(A)	矩阵 A 的分解
eig(A)	矩阵 A 的特征值分解
hess(A)	矩阵 A 的形式
lU(A)	矩阵 A 的分解
null(A)	由奇异值分解得出的矩阵 A 的零空间的标准正交基
orth(A)	矩阵 A 行向量的标准正交基
pinv(A)	求矩阵 A 的伪逆
qr(A)	矩阵 A 的 QR 正交三角分解
qz(A)	矩阵 A 的 QZ 分解，用于广义特征值
rref(A)	将矩阵 A 转换为逐行递减的阶梯阵
rsf2csf(V,D)	实数块对角形转换成复数对角形
schur(A)	矩阵 A 的分解
subspace(A)	计算出 A、B 形成的子空间的夹角
svd(A)	方阵 A 的奇异值分解

4. 稀疏矩阵

在实际应用中，常常会碰到只包含几个非零元素的矩阵，即稀疏矩阵。如果稀疏矩阵的规模很大，那么，无论是对零元素进行存储还是计算，都会对计算机资源造成极大的浪费。为了节省计算机的存储空间，通常只存储非零元素以及表示这些元素的行和列位置的两个下标数组。同样，为避免对零元素的代数运算，开发了特殊的算法来求解典型的矩阵问题。

MATLAB 的稀疏矩阵函数如表 2.9 所示。

表 2.9 稀疏矩阵函数

函数名称	功 能
colmmd	列最小度
colperm	基于非零元统计进行列排序
condest	求 1–范数矩阵条件
dmperm	杜尔梅奇门德尔松分解
find	求非零项的下标

续表

函数名称	功　能
full	将稀疏矩阵变成满阵形式
gplot	按图论画出稀疏矩阵
nnz	非零元素的数目
nonzeros	寻找非零元素
normest	估计 2–范数
nzmax	为非零元素分配的内存数
randperm	随机排列向量
spalloc	给非零元素分配内存
sparse	创建稀疏矩阵
spaugment	形成最小二乘增广系统
spconvert	将稀疏矩阵变换成外部格式
spdiags	从对角阵生成稀疏矩阵
speye	稀疏单位阵
spones	用 1 代替非零元素
spparms	设置稀疏矩阵例行参数
sprandn	随机稀疏矩阵
sprandsym	随机稀疏对称阵
sprank	稀疏结构的秩
spy	显示稀疏结构
symmd	最小对称度
symrcm	逆思尔麦基序

稀疏矩阵创建命令是 sparse，sparse(i,j,s) 创建的稀疏矩阵第 k 个非零元素是 s(k)，s(k) 位于第 i(k)行第 j(k)列。

稀疏带状矩阵创建命令为 spdiags，外部数据转换为稀疏矩阵的命令为 spconvert，稀疏矩阵的显示命令为 spy。

spy(S,'LineSpec',markersize)中的 LineSpec 字符串控制线型及其颜色等，参数 markersize 指定点的大小。

举例如下：

```
%用两种不同方式创建三对角稀疏矩阵。
n=5;
SM1=sparse(1:n,1:n,-2*ones(1,n),n,n,n);
SM2=sparse(2:n,1:n-1,ones(1,n-1),n,n,n-1);
S1=SM1+SM2+SM2'
e=ones(n,1);
S2=spdiags([e,-2*e,e],[-1,0,1],n,n)
SF=full(S1)  %将稀疏矩阵变成满阵形式。
S1 =
   (1,1)        -2
```

```
   (2,1)          1
   (1,2)          1
   (2,2)         -2
   (3,2)          1
   (2,3)          1
   (3,3)         -2
   (4,3)          1
   (3,4)          1
   (4,4)         -2
   (5,4)          1
   (4,5)          1
   (5,5)         -2
S2 =
   (1,1)         -2
   (2,1)          1
   (1,2)          1
   (2,2)         -2
   (3,2)          1
   (2,3)          1
   (3,3)         -2
   (4,3)          1
   (3,4)          1
   (4,4)         -2
   (5,4)          1
   (4,5)          1
   (5,5)         -2
SF =
   -2     1     0     0     0
    1    -2     1     0     0
    0     1    -2     1     0
    0     0     1    -2     1
    0     0     0     1    -2
%稀疏矩阵的图形显示。
clear all,n=200;                              %给出矩阵的阶数。
A=sprandsym(n,0.015,0.1,1);                   %建立(200*200)随机正定稀疏矩阵。
spy(A,'b',10),
title('Spy plot of matrix A')
```

图 2.1 稀疏矩阵结构

运行结果如图 2.1 所示。

2.2.4 关系运算和逻辑运算

除了基本的数学运算外，MATLAB 还支持关系运算和逻辑运算，其目的是提供求解真假命题的答案，这便于控制基于真假命题的一系列 MATLAB 命令的流程或执行的顺序。

在执行关系及逻辑运算时，MATLAB 将输入的不为零的数值都视为真(True)，而为零的数值则视为假(False)。运算的输出值将判断为真者以 1 表示，而判断为假者以 0 表示。

MATLAB 提供的关系运算符，如表 2.10 所示。

表 2.10 关系运算符

关系运算符	功 能	关系运算符	功 能
<	小于	>=	大于或等于
<=	小于或等于	==	等于
>	大于	~=	不等于

在 MATLAB 中，关系运算符能用来比较两个同样大小的数组，或用来比较一个数组和一个标量。

若是数组 A 和数组 B 比较，则两个数组按对应元素逐个进行比较，并产生一个同样规模的逻辑数组，其元素仅由 0 和 1 构成，比较的结构为假时，值为 0；为真时，值为 1。

若是数组 A 和标量 a 比较，则数组中的每一个元素都与 a 比较，结果同样产生一个与数组 A 同样规模的逻辑数组。

下面是几个对数组进行关系运算的例子。

```
a=1:9
a =
     1    2    3    4    5    6    7    8    9
b=5-a
b =
     4    3    2    1    0   -1   -2   -3   -4
tf=a>4
tf=
     0    0    0    0    1    1    1    1    1
tf=a==b
tf=
     0    0    0    0    0    0    0    0    0
tf=b-(a>2)
tf=
     4    3    1    0   -1   -2   -3   -4   -5
```

1. 逻辑运算符

MATLAB 提供了 4 个逻辑运算符，如表 2.11 所示，逻辑运算只能在相同规模的数组之间进行。

表 2.11 逻辑运算符

逻辑运算符	功 能
&	逻辑与
|	逻辑或
~	逻辑非
xor	逻辑异或

下面对这 4 个逻辑运算符作一说明：

假设已有标量 a 以及相同规模的数组 A 和数组 B。

- 逻辑与

  A&B 的结果为一个逻辑数组，结果数组的元素为 A 和 B 相应元素进行比较的逻辑与值。

  A&a 的结果也是一个同样规模的数组，其元素为标量与数组的每一个元素进行逻辑与运算的结果值。
- 逻辑或

  A|B 的结果为一个数组，结果数组的元素为 A 和 B 相应元素进行比较的逻辑或值。
- 逻辑非

  ~A 的结果为一个数组，结果数组的元素为 A 相应元素的逻辑非值。
- 逻辑异或

  xor(A,B) 的结果为一个数组，结果数组的元素为 A 和 B 相应元素进行比较的逻辑异或值。

以下有几个例子：

```
a=1:9              %定义一个数组。
a=
     1     2     3     4     5     6     7     8     9
tf=~(a>4)           %逻辑非。
tf=
     1     1     1     1     0     0     0     0     0
tf=(a>2)&(a<6)     %逻辑与。
tf=
     0     0     1     1     1     0     0     0     0
A=[1,0;0,5];         %定义矩阵。
B=ones(2);
A&B                %逻辑与。
ans=
     1     0
     0     1
A|B                 %逻辑或。
ans=
     1     1
     1     1
xor(A,B)            %逻辑异或。
ans=
     0     1
     1     0
~A                  %逻辑非。
ans=
     0     1
     1     0
x=2;                 %定义一个标量。
A&x
ans=
     1     0
     0     1
```

下在的例子是利用关系及逻辑运算产生的不连续的信号。

```
x=linspace(0,10,100);                 % 产生数据。
y=sin(x);                             % 产生 sine 函数。
z=(y>=0).*y;                          % 将 sin(x) 的负值设为零。
z=z+0.5*(y<0);                        % 再将上式的值加上 0.5。
z=(x<8).*z;                           % 将大于 x=8 以后的值设为零。
plot(x,z)                             % 以数组 x 为横坐标值，数组 z 为纵坐标值作图。
xlabel('x'),ylabel('z=f(x)')          %标记坐标轴。
title('A discontinuous signal')%标记图形名称。
```

生成的结果如图 2.2 所示。

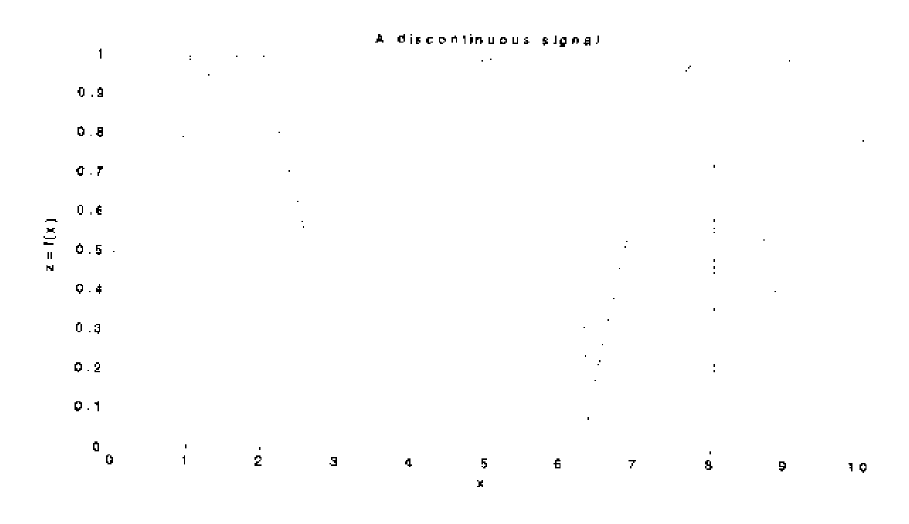


图 2.2 一个不连续的信号

2. 关系和逻辑函数

除了上述的关系和逻辑运算符之外，尚有以下的逻辑关系函数，其使用方式如表 2.12 所示。

表 2.12 逻辑关系函数

函数名	功 能
all	如果在一个向量中，所有元素非零，返回真值 1；矩阵中的每一列所有元素非零，返回真值 1
any	如果在一个向量中，任何元素是非零，返回真值 1；矩阵中的每一列有非零元素，返回真值 1
exist	变量或函数被定义，返回真值 1
finite	元素有限，返回真值 1
find	求非零元素下标(常用于稀疏矩阵)。作用于向量时，返回向量非零元素下标；作用于矩阵时，返回向量非零元素的行列下标
isempty	参数为空，返回真值 1
isglobal	参数为一个全局变量，返回真值 1
ishold	当前绘图保持状态是 on，返回真值 1

续表

函数名	功　　能
isieee	计算机执行 IEEE 算术运算，返回真值 1
isinf	元素无穷大，返回真值 1
isletter	元素为字母，返回真值 1
isnan	元素为不定值，返回真值 1
isreal	参数无虚部，返回真值 1
isspace	元素为空格字符，返回真值 1
issparse	矩阵为稀疏矩阵，返回真值 1
isstr	参数为一个字符串，返回真值 1

举例说明如下：

- 函数 all、any、exist 的使用

```
%定义 3 个矩阵。
A=[0,1;0,2;3,4]
A =
     0     1
     0     2
     3     4
B=[2,4;6,8;0,0]
B =
     2     4
     6     8
     0     0
C=[1,2;6,8;9,3]
C =
     1     2
     6     8
     9     3
%函数 all 的使用。
all(A)
ans =
     0     1
all(B)
ans =
     0     0
all(C)
ans =
     1     1
%函数 any 的使用
D=[0,1;0,2]
D =
     0     1
     0     2
any(A)
ans =
     1     1
```

```
any(B)
ans =
     1     1
any(D)
ans =
     0     1
%函数 exist 的使用,接上例。
exist('A')    %A 已定义。
ans =
     1
exist('E')    %E 未定义。
ans =
     0
```

- 函数 finite、isempty 的使用

```
%定义一个数组。
a=[1,inf,nan]
a =
     1   Inf   NaN
finite(a)        %inf、NaN 均为无限值。
ans =
     1     0     0
isempty(a)      %数组 a 中有元素。
ans =
     0
```

- 函数 isglobal 的使用

```
%定义一个变量 v 和一个全局变量 av。
v=1;
global av
av=2;  %在工作空间中显示类型 class 为 double array(global)。
%调用函数 isglobal。
isglobal(v)
ans =
     0
isglobal(av)
ans =
     1
```

- 函数 isstr、isreal 的使用

```
%定义字符串变量 sv 和实数变量 rv。
sv='abcd';
rv=3.45;
%调用函数 isstr、isreal。
isstr(sv)
ans =
     1
isstr(rv)
ans =
     0
```

```
isreal(rv)
ans =
     1
isreal(4*i)  %4*i 为一虚数。
ans =
     0
```

- 函数 isinf、isnan 的使用

```
%定义一个数组。
a=[1,inf,nan]
a =
     1   Inf   NaN
%调用函数 isinf 及 isnan。
isinf(a)
ans =
     0     1     0
isnan(a)
ans =
     0     0     1
```

3. 非数和空数组

非数 NaN 是 MATLAB 系统规定的一个特殊变量，即不定量；空数组([ ])是系统规定的一种特殊数组。由于其特殊性，在 MATLAB 中必须对其作特殊处理，尤其是在关系和逻辑表达式中使用时。

- 非数 NaN 的产生和性质
  - ◆ 非数的产生

```
a=0/0,b=0*log(0),c=inf.inf
Warning: Divide by zero.
a =
   NaN
Warning: Log of zero.
b =
   NaN
c =
   NaN
```

  - ◆ 非数的传递性

```
0*a,sin(a)
ans =
   NaN
ans =
   NaN
```

  - ◆ 非数的不可比较性

```
a==nan                %该命令想计算“a 等于非数吗？”。但不能给出正确的判断结果。
ans =
     0
```

◆ 非数不能进行关系运算

```
a~=nan              %该命令想计算"a 不是非数吗？"，也不可能给出正确的判断结果。
a==b                %两个非数不存在"等"与"不等"的概念。
b>c                 %两个非数不能比较大小。
ans =
     1
ans =
     0
ans =
     0
```

◆ 数的属性判断

```
class(a)            %数据类型归属。
isnan(a)            %该命令是唯一能正确判断非数的命令。
ans =
    double
ans =
     1
```

● 非数元素的寻访

```
%创建带非数的二维数组。
rand('state',0) %将随机发生器置 0。目的是使读者便于把自己运算结果与本书对照。
R=rand(2,5);
R(1,5)=NaN;
R(2,3)=NaN;
R =
    0.9501    0.6068    0.8913    0.4565       NaN
    0.2311    0.4860       NaN    0.0185    0.4447
isnan(R)                  %对数组元素是否非数进行判断。
ans =
     0     0     0     0     1
     0     0     1     0     0
%找出非数元素的位置标识
Linear_index=find(isnan(R))       %非数的"单下标"标识。
[r_index,c_index]=ind2sub(size(R),Linear_index);    %转换成"全下标"标识。
disp('r_index c_index'),disp([r_index c_index])
Linear_index =
     6
     9
r_index   c_index
     2     3
     1     5
```

● 空数组

◆ 创建空数组的几种方法

```
a=[],b=ones(2,0),c=zeros(2,0),d=eye(2,0),f=rand(2,3,0,4)
a =
     []
b =
```

```
   Empty matrix: 2-by-0
c =
   Empty matrix: 2-by-0
d =
   Empty matrix: 2-by-0
f =
   Empty array: 2-by-3-by-0-by-4
```

◆ 空数组的属性

```
class(a)            % "空" 的数据类别。
isnumeric(a)        %是数值数组类吗?
isempty(a)          %唯一可正确判断数组是否 "空" 的命令。
ans =
double
ans =
     1
ans =
     1

which a             %变量 a 是什么。
ndims(a)            %数组 a 的维数。
size(a)             %a 数组的大小。
a is a variable.
ans =
     2
ans =
     0     0
```

◆ 空数组不具备一般的传递性。

```
b_c1=b.*c           %两个空阵的点乘。
b_c2=b'*c           %矩阵乘一。注意：生成矩阵为 0-by-0，故 "空" 。
b_c3=b*c'           %矩阵乘二。注意：生成矩阵为 2-by-2。
b_c1 =
   Empty matrix:2-by.0
b_c2 =
     []
b_c3 =
     0     0
     0     0
```

◆ 空数组的比较要谨慎

```
a==b          %结果解释不合理。
Warning: [] == X is technically incorrect. Use isempty(X) instead.
ans =
     0
b==c          %结果可合理解释为 "无法比较" 。
ans =
   Empty matrix: 2-by-0
c>d           %结果可合理解释 "无法比较" 。
ans =
```

```
   Empty matrix: 2-by-0
a==0        %结果可解释为“不等于”。
Warning:Future versions will return empty for empty==scalar
comparisons.
ans =
     0
a~=0     %结果解释为“是不等”。
Warning:Future versions will return empty for empty~=scalar
comparisons.
ans =
     1
```

◆ 没有空数组参与运算时，结果中的“空”有合理的解释

```
A=reshape(-4:5,2,5) %创建一个数值数组 A。
A =
    -4    -2     0     2     4
    -3    -1     1     3     5
L2=A>10               %检查 A 中大于 10 的元素位置。
find(L2)                      %找出 L2 逻辑数组中非 0 元素的“单下标”标识。
L2 =
     0     0     0     0     0
     0     0     0     0     0
ans =
     []
```

◆ 空数组用于子数组的删除和大数组的维数收缩

```
A(:,[2,4])=[]              %删除 A 的第 2、4 列。
A =
    -4     0     4
    -3     1     5
```

2.2.5　字符与字符串的基本运算

MATLAB 有强大的字符处理能力，特别是在引入符号计算(Symbolic Math)工具包以后。对于编程语言来讲，字符处理是必不可少的。本节将介绍字符和字符串的基本运算。关于符号运算，将在 2.2.6 节中详细介绍。

在 MATLAB 中关于字符串有以下几点规则：

● 在 MATLAB 中所有字符串都用单引号界定后输入或赋值(yesinput 命令除外)。
如命令 s='Hello'的运行结果如下：

```
s=
Hello
```

● 字符串的每个字符(空格也是字符)都响应矩阵的一个元素。
如上述 s 变量是(1×5)的矩阵，这可用命令 size(s)查得。

● 字符以 ASCII 码贮存，用 abs 命令可看到字符的 ASCII 码值。
如运行 abs(s)可得到如下结果：

```
ans =
    72   101   108   108   111
```

- 用命令 setstr 可以实现 ASCII 码值向字符的转换。
- 字符变量也可以用方括号合并成更大的字符串。
  例如，运行命令 s=[s,'world']，可得到下面结果：

```
s=
Helloworld
```

- 用 eval/feval 函数将字符变量转换为宏功能。eval(t)/feval(t)就是运行包含在 t 中的内容。

```
%用 eval 函数产生 5 阶的 Hilbert 矩阵。
n=5;
t='1/(i+j.1)';
a=zeros(n);
for i=1:n
   for j=1:n
      a(i,j)=eval(t);
   end
end
a
a =
    1.0000    0.5000    0.3333    0.2500    0.2000
    0.5000    0.3333    0.2500    0.2000    0.1667
    0.3333    0.2500    0.2000    0.1667    0.1429
    0.2500    0.2000    0.1667    0.1429    0.1250
    0.2000    0.1667    0.1429    0.1250    0.1111
%feval 函数的使用
fun=['sin';'cos';'log'];
k=input('Choose function number:');
x=input('Enter value:');
feval(fun(k,:),x)
```

说明： feval 与 eval 不同，它以输入变量作为某个函数的函数名来执行，因此，可以使用 feval 和 input 命令从几个 M 文件定义的命令中选择一个。

常用字符串的操作函数及其说明如表 2.13 所示。

表 2.13 字符串操作函数

字符串操作函数	说　明
isstr	判断是否为字符
blanks	空白字符
deblank	移去空白字符
eval	运行字符串
strcmp	比较字符串
findstr	从一个字符串中寻找是否包含另一个字符串

续表

字符串操作函数	说　明
strrep	用一个字符串代替另一个字符串
upper	将字符串变为大写形式
lower	将字符串变为小写形式
abs	将字符串变为 ASCII 码值
setstr	将 ASCII 码值变为字符串
num2str	将数字变为字符串
int2str	将整数变为字符串
str2num	将字符串变为数字
str2mat	将字符变为文本矩阵
sprintf	将带格式的数字转变为字符串
sscanf	将字符串转变为带格式的数字
hex2num	将十六进制的字符串转变为 IEEE 浮点数
hex2dec	将十六进制的字符串转变为十进制数
dec2hex	将十进制数转变为十六进制的字符串

2.2.6　符号运算

MATLAB 的符号计算是以加拿大 Waterloo Maple 公司的 Maple V4 作为基本的符号计算引擎，借助于 Maple 已有的数据库，开发了实现符号计算的工具箱(Symbolic Math toolbox)，该工具箱有一百多个 M 文件，并且在 MATLAB 中可以通过 Maple.m 直接调用 Maple 的所有函数实现符号计算。

MATLAB 具有的符号数学工具箱与其他的所有工具不同，它用途广泛，而不是针对一些特殊专业或专业分支。另外，MATLAB 符号数学工具箱与其他工具箱的区别还在于它使用字符串进行符号分析，而不是基于数组的数值分析。

符号数学工具箱是操作和解决符号表达式的符号数学工具箱(函数)集合，包括复合、简化、微分、积分以及求解代数方程和微分方程的工具。另外有一些用于线性代数的工具，求解逆、行列式、正则型式的精确结果，找出符号矩阵的特征值而无由数值计算引入的误差。工具箱还支持可变精度运算，即支持符号计算并能以指定的精度返回结果。

1.　符号变量、符号表达式、符号方程和符号矩阵

在数值计算(包括输入、输出及中间计算)在内的过程中，所运作的变量都被赋了值的数值变量。而在符号计算的整个过程中，所运作的是符号变量。因此，要想掌握符号运算，首先必须弄清楚什么是符号变量，什么是符号表达式，如何创建它们，如何生成符号函数。接下来分别加以说明。

- 符号表达式和符号方程的创建

在符号计算中创建了一个新的数据类型：sym 类，即符号类，该类型的实例就是符号

对象，在符号计算工具箱内，用符号对象表示符号变量和符号矩阵等，并构成符号表达式和符号方程。

符号表达式是数字、函数、算子和变量的 MATLAB 字符串或字符串数组，不要求变量有预先确定的值，符号方程式是含有等号的符号表达式。符号算术是使用已知的规则和给定符号恒等式求解这些符号方程的实践，它与代数和微积分所学到的求解方法完全一样。符号矩阵是数组，其元素是符号表达式。

符号计算中出现的数字也都是当作符号处理的。MATLAB 在内部把符号表达式表示成字符串，以与数字变量或运算相区别；否则这些符号表达式几乎完全类似于基本的 MATLAB 命令。

符号表达式和符号方程是两种不同的对象。它们的区别在于前者不包含等号，而后者必须带等号。但这两种对象的创建方式相同，它们最简单和最常用的创建方式与在 MATLAB 创建字符串变量的方式几乎相同。例如：

```
%符号表达式和符号方程赋给符号变量举例。
g='1/(2*x^n)'    %所创建的函数 1/(2x^n) 赋给变量 g。
f='b*x+c=0'  %所创建的方程 bx+c=0 赋给变量 f。
si='sin(x)=0.02'     %所创建的方程赋给变量 si。
```

第一个等号右边的部分，既可以是符号表达式也可以是符号方程。以上 3 个例子都是把创建的符号表达式或符号方程赋给了符号变量。引入符号变量的目的是为以后调用方便，但这并不是必须的。事实上，符号表达式可以直接参与运算。

符号表达式和符号方程对空格都非常敏感。因此，在创建符号表达式时，不要在字符间任意乱加“修饰性”空格符。

以上将符号表达式和符号方程赋给符号变量就是用直接赋值法创建符号变量。因为符号矩阵是数组，其元素是符号表达式，所以也可以直接给数组赋值创建符号矩阵，请看下例：

```
matsym=['[a1,a2,a3]'; '[b1,b2,b3]'; '[c1,c2,c3]']     %创建符号矩阵 matsym。
matsym =
    [a1,a2,a3]
    [b1,b2,b3]
    [c1,c2,c3]
```

注意： MATLAB 要求符号矩阵的每列包含的字符个数必须相等，如果不相等，MATLAB 会显示错误。而且，符号矩阵每一行的两端都有方括号，这是与 MATLAB 字符串矩阵的一个非常重要的区别。

- 符号表达式中独立变量的确定

在符号计算的整个过程中，运作的是符号变量，当字符表达式含有多于一个的变量时，只有一个变量是独立变量。如果在前面的命令行中没有定义哪一个变量是独立变量，MATLAB 将基于以下规则选择一个：

- 在符号表达式中默认的独立变量是唯一的。MATLAB 对单个英文小写字母(除 i,j 外)进行搜索，且以 x 为首选独立变量。如果字符不是唯一的，就选择在字母顺序中最接近 x 的字母。如果有相连的字母，则选择在字母表中较后

的那一个。例如：

在表达式'3*y+z'中，独立变量是'y'；在表达式'sin(a*t+b)'中，独立变量是't'；在表达式'sqrt(i*alpha)'中，独立变量是'alpha'。

在表达式'sin(pi/4).cos(3/5)'中，独立变量是'x'，因为此式是一个符号常数而无符号变量。可利用函数 symvar 询问 MATLAB 认为在符号表达式中哪一个变量是独立变量。例如：

```
symvar('a*x+y')    %返回默认的独立变量，这里是 x。
```

- ◆ 如果 symvar 利用规则不能找到一个默认独立变量，它便假定无独立变量并返回 x。这一结论对含有由多个字母组成的变量(如 alpha 或 t3 的表达式)或不含变量的符号常数均成立。如果需要，大多数命令都使用用户选项以指定独立变量。

- 符号变量、符号矩阵的创建和修改

  除可以用直接赋值法创建符号变量和符号矩阵外，MATLAB 还提供了 sym()函数来创建符号变量和符号矩阵，其使用格式有以下几种。

  - ◆ S=sym(A)　由 A 创建符号类对象 S，如果 A 是数值矩阵，则 S 用数值的符号表示。
  - ◆ x=sym('x')　创建符号变量 x。
  - ◆ x=sym('x','real')　设定 x 为实变量。
  - ◆ x=sym('x','unreal')　unreal 设定符号变量 x 没有附加属性。
  - ◆ S=sym(A,flag)　此处参数 flag 可以是'f'、'r'=、'e'或'd'，分别代表浮点数、有理数、机器误差和十进制数。

下面是该函数应用的例子。

```
%先看参数 real 和 unreal 的不同。
x=sym('x','real');
x1=conj(x)
x1=
x
y=sym('y','unreal')
y=
y
y1=conj(y)
y1=
    conj(y)
%下例创建符号矩阵 mym。
mym=sym('[a1,a2,a3;b1,b2,b3;c1,c2,c3]')
mym =
[ a1, a2, a3]
[ b1, b2, b3]
[ c1, c2, c3]
%下例构造一个数值表示的符号变量。
r=sym('(1+sqrt(5))/2')
r =
(1+sqrt(5))/2
```

```
%用该符号变量 r 进行计算。
fun=r^2+r+1
fun =
(1/2+1/2*5^(1/2))^2+3/2+1/2*5^(1/2)
```

在数值计算中，通过一个命令就可实现对矩阵任何一个子阵的引用和修改。但在符号计算中，引用和修改只能对符号矩阵的元素一个一个进行。用函数 sym(S,i,j)可以取符号矩阵的指定元素，用 sym(S,i,j,'expr')可以修改指定的字符矩阵元素。

请看下面创建数值符号矩阵并取矩阵的元素的实例。

```
mat=magic (3)
mat =
     8     1     6
     3     5     7
     4     9     2
mats=sym(mat) %把数值矩阵转换为符号矩阵.
mats =
[ 8, 1, 6]
[ 3, 5, 7]
[ 4, 9, 2]
```

把这个命令与 MATLAB 中许多数值特殊矩阵生成命令(如 eye,ones,zeros)配合使用，可以产生许多特殊的符号矩阵。

```
s=sym(mats,2,2)
s=
    5
```

当要说明的符号变量较多时，可以使用 syms()函数，该函数调用格式 y 的方式有以下几种：

- syms var1 var2...　　同时说明 var1、 var2 等为符号变量。
- syms var1 var2...real
- syms var1 var2...unreal

其中，参数 real 和 unreal 说明这些变量是否为纯实变量。

举例说明如下：

```
a=sym('a');
b=sym('b');
c=sym('c');
x=sym('x');
V=sym(a*sin(b*x+c))
V =
a*sin(b*x+c)
```

前面 4 个对变量 a、b、c 及 x 的说明可用下句代替：

```
syms a,b,c,x
V=sym(a*sin(b*x+c))
V =
a*sin(b*x+c)
```

如果不能确认或者要从一个字符串中寻找变量，则可以用 symvar()函数，该函数可以找出字符串中的变量。调用格式如下：

```
symvar(S)
```

举例如下：

```
var=symvar('sin(2*pi*t+a)')
var =
    'a'
    't'
```

也可以用 findsym()函数查找字符串中的符号变量，该函数的调用格式分为以下两种：

- ◆ r=findsym(S) 查找字符串或矩阵 S 中的符号变量，结果按字母顺序排列，如果没有变量，返回空值。
- ◆ r= findsym(S,n) 参数 n 指定按字母顺序排列的最接近的符号变量。

举例说明如下：

```
syms x y z
r= findsym(x+i*y.j*z)
r =
x, y, z
r= findsym(x+i*y.j*z,2)
r =
    x,y
```

与函数 sym()相反，numeric()函数可以把符号常数转化为数值进行计算，恰好是函数 sym 的逆运算。请看该函数应用的例子。

```
r=sym('(1+sqrt(5))/2')    %黄金分割比。
r =
(1+sqrt(5))/2
fun=r^2+r+1
fun =
(1/2+1/2*5^(1/2))^2+3/2+1/2*5^(1/2)
numeric(fun)
ans =
    5.2361
```

2. **符号函数**

只有符号变量还不能解决复杂问题，很多时候还需要符号函数，MATLAB 提供了以下两种生成符号函数的方法。

- 直接用符号表达式生成符号函数

  直接用含有符号变量的符号表达式生成函数，一旦建立了符号函数就可以对其进行符号运算。请看下面的例子。

```
syms x y
fun=sqrt(x^2-2*x*y+y^2)
fun =
((-x+y)^2)^(1/2)
```

```
t=sin(x)
t =
sin(x)
res=diff(t)
res =
cos(x)
```

- 用 M 文件生成符号函数

  对于复杂的或者常用的符号函数，可以用 M 文件生成。下面是用 M 文件生成符号函数的例子。关于 M 文件的概念，将在第 3 章 MATLAB 语言结构与编程中详细介绍。

```
% MATLAB 的自建函数。
funtion y=sinc(x)
%SINC Sin(pi*x)/(pi*x) function
y=ones(size(x))
i=find(x);
y(i)=sin(pi*x(i))./(pi*x(i));
```

3. 符号函数运算

创建了一个符号表达式后，有时又想用某些方式改变它，也许希望提取表达式的一部分、合并两个表达式或求得表达式的数值，这时就可以利用一些符号函数来帮助我们完成这些任务。

所有的符号函数(例外的情况很少)都作用到符号表达式和符号数组，并返回符号表达式或数组。其结果有时可能看起来像一个数字，但事实上它是一个内部用字符串表示的符号表达式。在 MATLAB 的数值计算中，矩阵的加、减、乘、除等运算的操作命令都很直观简单。在符号计算中，情况就不同了，所有涉及符号计算的操作都要借助于专用函数来进行。

以下对符号矩阵适用的命令，都适用于符号表达式。当然，这些命令对符号方程是没有意义的。

- 基本运算

  符号矩阵的基本运算有加、减、乘、除、乘幂运算，实现符号矩阵加、减、乘、除、逆、幂的命令分别是：

  symadd(A,B)　　给出两个符号矩阵的和。

  symsub(A,B)　　给出两个符号矩阵的差。

  symmul(A,B)　　给出两个符号矩阵的积。相乘规则是：符号表达式可以与符号矩阵相乘；两个内维数相同的符号矩阵可以相乘。

  symdiv(A,B)　　给出两个符号矩形的商。A 可以是符号表达式或 m×n 维符号，矩阵 B 可以是符号表达式或 n×n 维满秩符号阵，symmul(A, inverse 节(B))与 symdiv(A,B)等价。

  inverse(B)　　符号矩阵 B 的除 inverse(B)和 symdiv('1',B)等价。

  sympow(S,p)　　求幂运算 sym(S)^sym(p)。若 S 为标量符号表达式，p 可为标量符号或数值表达式；若 S 为符号方阵，p 必须为整数。

举例如下：

```
syms a b
res1=symadd('cos(a)','b')
res1 =
cos(a)+b
res2=symsub(res1,'b')
res2 =
cos(a)
res3=symdiv('x^3+5*x','x+5')
res3 =
(x^3+5*x)/(x+5)
res4=sympow('x','y')
res4 =
x^y
```

- 符号矩阵的综合运算命令

  除了前面几小节介绍的加、减、乘、除、逆、幂等单种符号计算命令外，在符号数学工具包中还有一个综合运算命令 symop()。符号矩阵的综合运算命令为 symop(s1,s2,s3,...)，s1,s2,s3,...分别是符号矩阵或数值矩阵或'+', '.', '*', '/', '^', '('和')'。举例如下：

```
f='x^2';
g='sin(x)';
ressym=symop(f,'*','cos(x)','+','-','3')  %使用综合运算命令。
ressym =
x^2*cos(x).3
```

注意：当综合运算命令中同时存在符号表达式和符号矩阵时，所得结果不一定正确；当综合运算命令中的括号()两侧同时存在乘、除运算时，所得结果也不一定正确。

- 因式分解、展开和简化

  对符号表达式、符号矩阵可以进行因式分解、展开和简化等操作，运算命令如下：

```
factor(S)           %对 S 进行因式分解操作。S 可以是单个表达式、有理分式及方程矩阵。
expand(S)           %对符号矩阵进行展开。
collect(S)          %把 S 符号矩阵元素中的同幂项系数进行合并。
collect(S,v)        %把 S 符号矩阵元素中默认变量的同幂项系数进行合并。
simple(S)           %显示 S 被简化的过程，并给出最简结果。
[R,HOW]=simple(S)   %不显示简化过程。R 为简化结果，HOW 为特别的简化操作。
```

举例如下：

在 factor 命令的使用中除 x 外不含其他自由变量的情况。

```
syms a x;f1=x^4-5*x^3+5*x^2+5*x-6;factor(f1)
ans =
    (x-1)*(x-2)*(x-3)*(x+1)
```

含其他自由变量的情况之一。

```
f2=x^2-a^2;factor(f2)
```

```
ans =
(x-a)*(x+a)
```

对正整数的质数分解。

```
factor(1025)
ans =
    5     5    41
%expand(S) 对符号矩阵进行展开。
GS=sym('[sin(x+y)/cos(2*x);exp(x-y);log(a*b^2/c);(s+1)*(s+3)/((s+2)*
(s+4))]');
expand(GS)
ans =
[ 1/(2*cos(x)^2-1)*sin(x)*cos(y)+1/(2*cos(x)^2-1)*cos(x)*sin(y)]
[                                              exp(x)/exp(y)]
[                                               log(a*b^2/c)]
[            1/(s+2)/(s+4)*s^2+4/(s+2)/(s+4)*s+3/(s+2)/(s+4)]

%collect 按不同的方式合并同幂项。
EXPR=sym('(x^2+x*exp(-t)+1)*(x-exp(-t))');
expr1=collect(EXPR)                          %默认合并 x 同幂项系数。
expr2=collect(EXPR,'exp(-t)')         %合并 exp(-t)同幂项系数。
expr1 =
x^3+2*exp(-t)*x^2+(1+exp(-t)^2)*x+exp(-t)
expr2 =
x*exp(-t)^2+(2*x^2+1)*exp(-t)+(x^2+1)*x
```

%运用 simple 简化，简化 $f=\sqrt[3]{\frac{1}{x^3}+\frac{6}{x^2}+\frac{12}{x}+8}$ 。

```
syms x;f=(1/x^3+6/x^2+12/x+8)^(1/3);
g1=simple(f),g2=simple(g1)
g1 =
(2*x+1)/x
g2 =
2+1/x
```

- 符号矩阵分解

与数值计算一样，在符号计算中，符号矩阵分解也特别有用。计算符号矩阵维数、转置、行列式、特征值分解、奇异值分解、零空间和列空间分解的命令如下所示：

```
symsize(S)                %求 S 矩阵维数的命令。
transpose(S)              %求 S 的符号转置矩阵。
determ(S)                 %求 S 阵的行列式。
colspace(S)               %给出 S 列空间的基。
symsize(colsize(S),2)     %给出 S 的秩。
nullspace(S)              %给出 S 零空间的基。
symsize(nullspace(S),2)   %给出 S 的零比度。
[VE,E]=eigensys(S)        %VE 给出矩阵 S 的特征向量，E 给出 S 的特征值。
[VJ,J]=jordan(S)          %VJ 给出符号矩阵 S 的广义特征向量，J 给出相应的约当标准
形。
singvals(A)               %给出一般符号阵 A 的奇异值。
[U,S,V]=singvals(A)       %给出(不带自由变量的)A 阵的奇异分解三对组。
```

举例如下：

```
%无重根阵的分解。
D=sym('[1,-3;2,2/3]');
[VE,E]=eigensys(D)
VE =
[                    1,                      1]
[1/18-1/18*i*215^(1/2),1/18+1/18*i*215^(1/2)]
E =
[ 5/6+1/6*i*215^(1/2),                    0]
[                   0,5/6-1/6*i*215^(1/2)]
%有重根阵的分解。
C=sym('[1,1,2;0,1,3;0,0,2]');
[VJ,J]=jordan(C)
VJ =
[  5,  -5,-5]
[  3,   0,.5]
[  1,   0,0]
J =
[ 2,  0,  0]
[ 0,  1,  1]
[ 0,  0,  1]
```

- 符号微积分

  微分和积分是微积分学研究和应用的核心，并广泛应用在许多工程学科。MATLAB 符号工具能帮助解决以下问题：

  - 符号微分

  微分函数为 diff()，它可以对符号表达式求微分，也可以对数组求微分。举例如下：

  %求 $\frac{d}{dx}\begin{bmatrix}\frac{a}{t\cos x} & \frac{t^3}{\ln x}\end{bmatrix}$、$\frac{d^2}{dt^2}\begin{bmatrix}\frac{a}{t\cos x} & \frac{t^3}{\ln x}\end{bmatrix}$和$\frac{d^2}{dxdt}\begin{bmatrix}\frac{a}{t\cos x} & \frac{t^3}{\ln x}\end{bmatrix}$。

```
syms a t x;f=[a,t^3;t*cos(x), log(x)];
df=diff(f)                              %求矩阵 f 对 x 的导数。
dfdt2=diff(f,t,2)                       %求矩阵 f 对 t 的二阶导数。
dfdxdt=diff(diff(f,x),t)                %求二阶混合导数。
df =
[          0,          0]
[-t*sin(x),        1/x]
dfdt2 =
[   0, 6*t]
[   0,   0]
dfdxdt =
[        0,        0]
[ -sin(x),        0]
```

注意： 函数 diff()也用在计算数值向量或矩阵的数值差分。对于一个数值向量或矩阵 M，diff(M)计算 M(2:m,:) - M(1:m - 1,:)的数值差分。例如：

```
%数值向量的差分。
M=[(1:8).^2] %建立一个向量。
```

```
M =
     1     4     9    16    25    36    49    64
diff(M) %计算元素之间的差分。
ans =
     3     5     7     9    11    13    15
```

如果 diff 的表达式或可变量是数值，MATLAB 就非常巧妙地计算其数值差分；如果参量是符号字符串或变量，MATLAB 就对其表达式进行微分。

◆ 符号积分

积分函数为 int(f)，积分比微分复杂得多，积分或逆求导不一定以封闭形式存在；或者存在但软件也许找不到；或者软件可明显地求解，但超过内存或时间限制。当 MATLAB 不能找到逆导数时，它将返回未经计算的命令。举例如下：

%求$\int\begin{bmatrix} ax & bx^2 \\ \frac{1}{x} & \sin x \end{bmatrix}dx$。演示，积分命令对符号函数矩阵的作用。

```
syms a b x;f=[a*x,b*x^2;1/x,sin(x)];
disp('The integral of f is');pretty(int(f))
The integral of f is
                       [       2          3]
                       [1/2 a x    1/3 b x ]
                       [                   ]
                       [ log(x)     .cos(x) ]
```

%求$\int_0^x \frac{1}{\ln t}dt$。演示如何使用 mfun 命令获取一组积分值，命令 mfun 的使用可参看联机帮助。

◆ 求一般积分结果

```
F1=int('1/log(t)','t',0,'x')
F1 =
-Ei(1,-log(x))
```

◆ 利用 mfun 命令求 x=0.5 , 0.6 , 0.7 , 0.8 , 0.9 时的定积分

```
x=0.5:0.1:0.9
F115=-mfun('Ei',1,-log(x))
x =
    0.5000    0.6000    0.7000    0.8000    0.9000
F115 =
   -0.3787   -0.5469   -0.7809   -1.1340   -1.7758
```

%求积分$\int_1^2\int_{\sqrt{x}}^{x^2}\int_{\sqrt{xy}}^{x^2y}(x^2+y^2+z^2)dzdydx$。注意：内积分上下限都是函数。

```
syms x y z
F2=int(int(int(x^2+y^2+z^2,z,sqrt(x*y),x^2*y),y,sqrt(x),x^2),x,1,2)
VF2=vpa(F2)               %积分结果用 32 位数字表示。
F2 =
1610027357/6563700-6072064/348075*2^(1/2)+14912/4641*2^(1/4)+64/225*
2^(3/4)
VF2 =
224.92153573331143159790710032805
```

%利用 rsums 求 $S=\int_0^{0.5}\frac{1}{\ln t}dt$ 积分。命令 rsums 的使用可参看联机帮助。

```
syms x positive;px=0.5/log(0.5*x);rsums(px)
```

结果如图 2.3 所示。

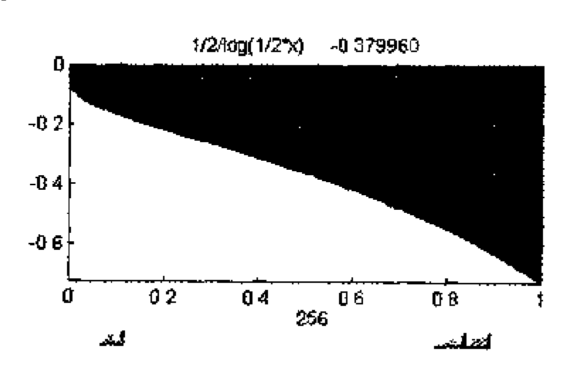


图 2.3　交互式近似积分

- 符号矩阵的代数运算

  除了上面介绍的几种微积分函数之外，还有一些函数在对符号矩阵进行处理时也会用到。本节着重介绍求符号和、符号导数、符号积分、泰勒级数以及符号雅可比矩阵。具体命令如下：

```
symsum(S,v)          %关于指定变量 v 对通项 S 求不定和。
symsum(S,v,a,b)      %指定变量 v 在[a,b]之间取值，对通项 S 求和。
taylor(F,v)          %求 F 对变量 v 的泰勒展开，6 阶小量以余项形式给出。
taylor(F,v,n)        %求 F 对变量 v 的泰勒展开，n 阶小量以余项形式给出。
jacobian(F,v)        %求 F 的雅可比矩阵，F 是向量函数，是指定变量构成的向量。
```

说明： 上述命令中的 v 可以省略。默认时，运算将对默认变量进行。

举例如下：

◆ 求无限项级数的和，$\sum_{t=0}^{t-1}\left[t \quad k^3\right]$，$\sum_{k=1}^{\infty}\left[\frac{1}{(2k-1)^2} \quad \frac{(-1)^k}{k}\right]$。

```
syms k t;f1=[t k^3];
f2=[1/(2*k-1)^2,(-1)^k/k];
s1=simple(symsum(f1))              %f1 的自变量被确认为 t。
s2=simple(symsum(f2,1,inf))        %f2 的自变量被确认为 k。
s1 =
[ 1/2*t*(t-1),k^3*t]
s2 =
[ 1/8*pi^2,-log(2)]
```

◆ 求 $f=\begin{bmatrix} x_1e^{x_2} \\ x_2 \\ \cos(x_1)\sin(x_2) \end{bmatrix}$ 的雅可比(*jacobian*)矩阵。

```
syms x1 x2 x3;
f=[x1*exp(x2);x2;cos(x1)*sin(x2)];
```

```
v=[x1 x2];
fjac=jacobian(f,v)
fjac =
[          exp(x2),        x1*exp(x2)]
[               0,                  1]
[-sin(x1)*sin(x2),cos(x1)*cos(x2)]
```

4. 符号代数方程求解

本部分将介绍线性方程组的符号解和一般代数方程组的符号解两部分内容。

- 线性方程组的符号解

  本书只讨论 A 阵至少行满秩时的线性方程组 A*X=B 的解。下面是命令的使用格式：

```
X=linsolve(A,B)            %只给出特解。
[X,Z]=linsolve(A,B)        %将给出由 X 及 Z 构成的通解。
```

注意： 当矩阵 A 列数大于行数时，将给出解不唯一的警告提示。X 是方程组的一个特解，Z 是 A 阵零空间的基，通解形式是(X+p*Z)，式中 p 是可任意取值的自由参数。举例如下：

◆ 求$d+\frac{n}{2}+\frac{p}{2}=q, n+d+q-p=10, q+d-\frac{n}{4}=p, q+p-n-8d=1$线性方程组的解。

```
A=sym([1 1/2 1/2-1;1 1-1 1;1-1/4 -1 1;-8-1 1 1]);
b=sym([0;10;0;1]);X1=A\b
X1 =
    [1]
    [8]
    [8]
    [9]
```

◆ 求解上例前 3 个方程所构成的“欠定”方程组，并解释解的含义。

```
syms k
A2=A(1:3,:);X2=A2\b(1:3,1)          %求一个特解：最少非零元素的最小二乘解。
XX2=X2+k*null(A2)                       %构成通解。
A2*XX2                                        %验算。
X2 =
[0]
[8]
[4]
[6]
XX2 =
[k]
[8]
[4+4*k]
[6+3*k]
ans =
[0]
[10]
[0]
```

- 一般代数方程的解

  此处所讲的 solve 命令，可以解一般代数方程，包括线性方程、非线性方程和超越方程。当方程组不存在解析解时，如果又没有其他的自由参数，则 solve 将给出数值解。举例如下：

  - 求方程组 $uy^2+vz+w=0$，$y+z+w=0$ 关于 y,z 的解。

```
S=solve('u*y^2+v*z+w=0','y+z+w=0','y','z')
disp('S.y'),disp(S.y),disp('S.z'),disp(S.z)
S =
    y: [2x1 sym]
    z: [2x1 sym]
S.y
[-1/2/u*(-2*u*w-v+(4*u*w*v+v^2-4*u*w)^(1/2))-w]
[-1/2/u*(-2*u*w-v-(4*u*w*v+v^2-4*u*w)^(1/2))-w]
S.z
[1/2/u*(-2*u*w-v+(4*u*w*v+v^2-4*u*w)^(1/2))]
[1/2/u*(-2*u*w-v-(4*u*w*v+v^2-4*u*w)^(1/2))]
```

  - 用 solve 命令求 $d+\frac{n}{2}+\frac{p}{2}=q$，$n+d+q-p=10$，$q+d-\frac{n}{4}=p$ 构成的“欠定”方程组解。

```
syms d n p q;eq1=d+n/2+p/2.q;eq2=n+d+q.p.10;eq3=q+d.n/4.p;
S=solve(eq1,eq2,eq3,d,n,p,q);S.d,S.n,S.p,S.q
Warning: 3 equations in 4 variables.
> In E:\MATLAB6.1\toolbox\symbolic\solve.m at line 110
  In E:\MATLAB6.1\toolbox\symbolic\@sym\solve.m at line 49
ans =
     d
ans =
     8
ans =
     4*d+4
ans =
     3*d+6
```

  - 求 $(x+2)^x=2$ 的解。

```
clear all,syms x;s=solve('(x+2)^x=2','x')
s =
.69829942170241042826920133106081
```

5. 符号微分方程求解

接下来介绍常微分方程的计算机求解命令 dsolve。该命令的使用格式为：

```
[y1,y2,…]=dsolve(a1,a2,…,a12)
```

说明：

- 输入变量包括3部分内容：微分方程、初始条件和指定独立变量，其中微分方程是必不可少的输入内容。后两个内容视需要而定，可有可无。

- 关于指定独立变量的规定：若要指定独立变量，则总是由全部输入变量 a1，a2，…中的最后一个变量定义。若不对独立变量加以专门的定义，则本命令默认小写英文字母 x 或 t 为独立变量。
- 关于初始条件的规定：初始条件被写成 y(a)=b，Dy(c)=d 等。a、b、c、d 可以是变量使用符外的其他字符。当初始条件少于微分方程数时，在所得解中将出现任意常数符 C1，C2，……，解中任意常数符的数目等于所缺少的初始条件数。

求 $\frac{dx}{dt}=y,\frac{dy}{dt}=-x$ 的解。

```
S=dsolve('Dx=y,Dy=.x');
disp([blanks(12),'x',blanks(21),'y']),disp([S.x,S.y])
            x                     y
[  cos(t)*C1+sin(t)*C2,-sin(t)*C1+cos(t)*C2]
```

图 2.4 显示了微分方程 $y=xy'-(y')^2$ 的通解和奇解的关系。

```
y=dsolve('y=x*Dy.(Dy)^2','x')                          %求微分方程解。
clf,hold on,ezplot(y(2),[-6,6,-4,8],1)                 %画奇解。
cc=get(gca,'Children');                                %取奇解曲线的图柄。
set(cc,'Color','r','LineWidth',5)                      %把奇解画成粗红线。
for k=-2:0.5:2;
ezplot(subs(y(1),'C1',k),[-6,6,-4,8],1);
end         %画通解。
hold off,title('\fontname{隶书}\fontsize{16}通解和奇解')
y =
[ x*C1.C1^2]
[   1/4*x^2]
```

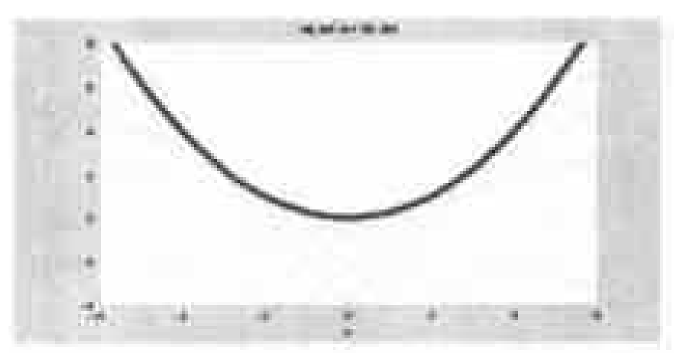


图 2.4 通解和奇解曲线

求解两点边值问题：$xy''+3'=x^2, y(1)=0, y(5)=0$。

```
y=dsolve('x*D2y.3*Dy=x^2','y(1)=0,y(5)=0','x')
y =
-1/3*x^3+125/468+31/468*x^4
```

求边值问题 $\frac{df}{dx}=3f+4g,\frac{dg}{dx}=-4f+3g,f(0)=0,g(0)=1$ 的解。

```
S=dsolve('Df=3*f+4*g,Dg=.4*f+3*g','f(0)=0,f(3)=1')
S.f,S.g
S =
```

```
    f: [1x1 sym]
    g: [1x1 sym]
ans =
     exp(3*t)*sin(4*t)/sin(12)/(cosh(9)+sinh(9))
ans =
     exp(3*t)*cos(4*t)/sin(12)/(cosh(9)+sinh(9))
```

6. 符号计算的扩展

相对MAPLE软件的2000余条符号计算命令而言，前面所介绍的内容仅利用了MAPLE最常用计算命令中的一部分。为了在MATLAB工作环境中，进一步利用MAPLE的其他符号计算能力，本节将介绍命令maple，该命令用于调动MAPLE的符号计算“引擎”和它庞大的函数库，并把最终的计算结果送到MATLAB工作区。

与前面一些符号计算命令相比，调用MAPLE要求MATLAB用户对MAPLE软件有更多的理解和认识。

- 直接调用MAPLE的符号计算能力

  在MATLAB环境下，为了实现对MAPLE绝大多数符号计算批令的调用，该符号数学工具包提供了一个通用命令maple。该命令的主要使用格式如下：

  maple(MAPLEStatement)　调用MAPLE中的完整命令MAPLEStatement。

  maple(a0,a1,a2……，a10)　调用MAPLE中名为a0的命令。

说明：a0是字符串；a1,a2……是a0命令使用格式中依次相应的输入变量。

接下来求递推方程 $f(n)=-3f(n-1)-2f(n-2)$ 的通解，具体操作如下：

(1) 调用格式一

```
gs1=maple('rsolve(f(n)=-3*f(n.1)-2*f(n.2),f(k));')
gs1 =
(2*f(0)+f(1))*(.1)^k+(-f(0)-f(1))*(.2)^k
```

(2) 调用格式二

```
gs2=maple('rsolve','f(n)=-3*f(n-1)-2*f(n.2)','f(k)')
gs2 =
(2*f(0)+f(1))*(.1)^k+(-f(0)-f(1))*(-2)^k
```

求 $f=xyz$ 的Hessian矩阵。

(1) 调用格式一

```
FH1=maple('hessian(x*y*z,[x,y,z]);')
FH1 =
matrix([[0,z,y],[z,0,x],[y,x,0]])
```

(2) 调用格式二

```
FH2=maple('hessian','x*y*z','[x,y,z]')
FH2 =
matrix([[0, z, y], [z, 0, x], [y, x, 0]])
```

(3) 把以上输出变成“符号”类

```
FH=sym(FH2)
FH =
[ 0, z, y]
[ z, 0, x]
[ y, x, 0]
```

求 $\sin(x^2+y^2)$ 在 $x=0, y=0$ 处展开的截断 8 阶小量的台劳近似式。

(1) 直接运行 mtaylor，展开失败。

```
TL1=maple('mtaylor(sin(x^2+y^2),[x=0,y=0],8)')
TL1 =
mtaylor(sin(x^2+y^2),[x = 0, y = 0],8)
```

(2) 必须先"读库"，才能得到正确的展开结果。

```
maple('readlib(mtaylor);');
TL2=maple('mtaylor(sin(x^2+y^2),[x=0,y=0],8)');
pretty(sym(TL2))
    2    2          6         2   4       4   2         6
 x +y-1/6x -1/2y  x -1/2y  x -1/6y
```

- MAPLE 的调试

MAPLE 提供了 2 个功能函数用以调试，它们是跟踪模式和输出返回值。

◆ 跟踪模式

命令 maple traceon 可以显示所有对 MAPLE 调用的中间过程，并将中间结果显示在屏幕上。例如，使用以下命令可将 exp(2*a)的过程显示出来。

```
maple traceon
a=sym('a');
exp(2*a)
statement=
(2)*(a);
result=
2*a
statement=
exp(2*a);
result=
exp(2*a)
ans=
exp(2*a)
```

使用 maple traceoff 命令可以关闭跟踪模式，使过程不再显示出来。

◆ 输出返回值

maple 函数可返回 result 和 status2 个参数。如果对 maple 过程的所有调用成功，则参数 result 的值为计算结果，而参数 status 的值为 0。如果调用失败，则 MAPLE 返回一个错误代码(一个正整数)，保存在 status 参数中，而在 result 参数中，则是对应的警告(错误)信息。例如：

```
syms a b c x
[result,status]=maple('disctim',a*x^2+b*x+c) %discrim函数计算一个多项式的
```

判别式，语法为 discrim(p,x)，此处忘记输入该函数的第 2 个参数。

```
result=
Error,(in discrim)  invalid  arguments
status=
2
```

如果调用正确，如下所示：

```
[result,status]=maple('disctim',a*x^2+b*x+c,x)
result=
-4*a*c+b^2
status=
0
```

7. 图形化的符号函数计算器

接下来要介绍 MATLAB 符号数学工具包所提供的第 3 种符号计算方式——图形化的函数计算方式。在 MATLAB 环境下，输入以下命令：

```
funtool
```

生成图形化的函数计算器如图 2.5 所示。

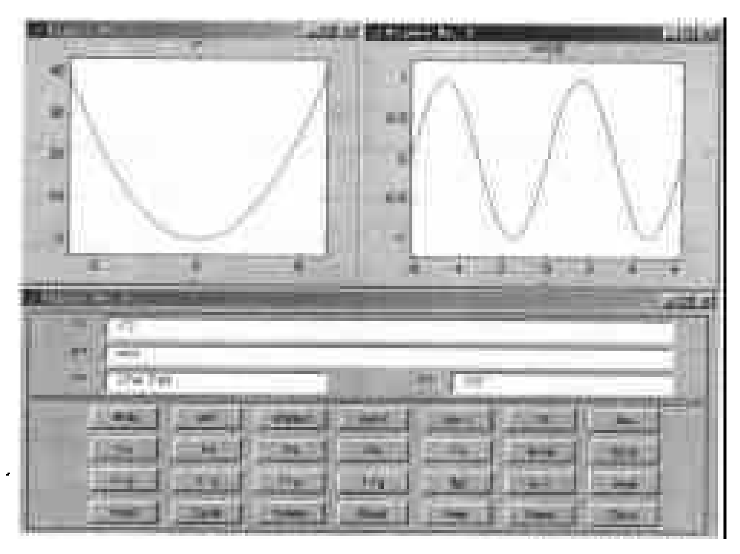


图 2.5 图形化的函数计算器

该图形化计算器由两个函数曲线视窗(Figure No.1 和 Figure No.2)和一个函数运算控制器(Figure No.3)构成。

- 函数曲线视窗的激活

  在任何时候，两个函数视窗中只有一个处于激活状态，在图 2.5 中，Figure No.2 正处于激活状态。单击视窗的任何位置，即可激活该函数视窗。

  在函数运算控制器上的任何操作，都只对激活的函数视窗起作用。换句话说，运算控制器上的不同操作，只有被激活的函数视窗中的图示曲线做相应的变化。

- 运算控制器上被控栏的操作

  被控栏是指函数运算控制器(Figure No.3)上半部分的 4 个文本框，即：f、g、x 和 a。其中 f、g 文本框分别显示相应视窗 Figure No.1 和 Figure No.2 中曲线的函数表

达式；x 文本框显示函数曲线视窗中作为横坐标变量的取值范围；a 文本框显示可能与函数运算的自由参数值。

被控栏中的内容有以下 3 种来源和操作方式：

- 在图形化函数计算器打开之前，若 MATLAB 工作区中已有同名字符串变量 f、g、x、a，那么这些变量的内容会被即将打开的图形化函数计算器所调用。反之，若 MATLAB 中没有已定义的同名变量，则新打开的图形化函数计算器将默认 f=x，g=1，x=[−2*pi,2*pi]，a=1/2。
- 已打开的图形化函数计算器上，除 x 文本框以外，被控栏 f、g 和 a 的内容可以用下述方法随时修改。把光标移到要修改的地方，通过键盘操作就可写入用户所希望的内容。结束以上操作之后，依次激活相应的函数曲线视窗，便可看到修改后所得的函数曲线。
- 当用运算控制器进行运算操作时，被激活的曲线视窗中的函数曲线和该视窗对应的被控栏中的函数表达式将实时反映运算结果。同时，MATLAB 的变量 f 和 g 中的内容也相应地改变。

● 单函数运算操作按钮

运算控制器上的第一排按钮都是单函数操作按钮。它们只对激活的那个函数进行以下运算，如表 2.14 所示。

表 2.14 单函数运用操作键按及其功能

按　钮	功　能
df/dx	求 F(x)相对于 x 的符号导数
int f	求 F(x)相对于 x 的符号积分
simple f	使 F(x)的表达式尽可能简化
num f	取 F(x)的分子表达式
den f	取 F(x)的分母表达式
1/f	求 1/f(x)
finv	求 F(x)的反函数，使 g(f(x))=x

如果在 int f 或是 fin v 操作中，得不到收敛的结果，那么相应的函数栏里会出现 NaN，表示运算失败。

● 函数和参数运算操作按钮

在运算控制器上的第 2 排按钮用来实现激活函数和自由参数 a 的操作，具体如表 2.15 所示。

表 2.15 函数和参数运算操作按钮

按　钮	功　能
f+a	计算 f(x)+a
f-a	计算 f(x).a
f*a	计算 af(x)

续表

按　钮	功　能
f/a	计算 f(x)/a
f ^ a	计算 $f^a(x)$
f(x+a)	计算 f(x+a)
f(x *a)	计算 f(ax)

- 两个函数间的运算操作按钮

  运算控制器上的第 3 排按钮用以实现两个函数间的运算。运算结果一方面显示在激活的函数视窗中，一方面显示在相应的被控函数栏里。运算操作按钮的具体功能如表 2.16 所示。

表 2.16　两个函数间的运算操作按钮及功能

按　钮	功　能
f+g	求 f(x)+g(x)
f-g	求 f(x)–g(x)
f*g	求 f(x)g(x)
f/g	求 f(x)/g(x)
f(g)	求复合函数 f(g (x))
g=f	用 f(x)取代 g(x)
swap	交换 f(x)、g(x)

- 辅助操作按钮

  运算控制器上的第 4 排按钮的功能如表 2.17 所示。

表 2.17　辅助操作按钮

按　钮	功　能
Insert	把当前 Figure No.1 视窗里的函数插入到内含的典型函数演示表中
Cycle	在 Figure No.1 视窗里依次演示内含的典型函数演示表中的函数曲线
Delete	从内含的典型函数演示表中删除当前 Figure No.1 视窗中的函数
Reset	把整个函数计算在重置成等待状态
Help	在主 MATLAB 窗里给出函数计算器的联机帮助
Demo	自动演示函数计算器的运算功能
Close	关闭函数计算器

第3章 MATLAB 6语言结构与编程

MATLAB 语言为解释性程序设计语言，语法简单、简洁有效。MathWork 公司将 MATLAB 语言称为第四代编程语言，它的编程效率比常用的 BASIC、C、FORTRAN 和 PASCAL 等语言要高得多，而且容易维护。

3.1 M文件的功能及形式

1. M文件的功能和特点

MATLAB 有两种常用的工作方式：一种是直接交互的命令行操作方式，另一种是文件的编程工作方式。在前一种工作方式下，MATLAB 被当作一种高级“数字演算和图示器”来使用。本章着重介绍与第 2 种工作方式有关的内容。

MATLAB 是一种强有力的操作环境，它集中了 MATLAB 所提供的完整而易于使用的编程语言。从形式上讲，MATLAB 程序文件是一个 ASCII 码文件(标准的文本文件)，扩展名为.m(M 文件的名称由此而来)。用任何字处理软件都可以对它进行编写和修改。从特征上讲，MATLAB 是解释性编程语言。其优点是语法简单，程序容易调试，人机交互性强。缺点是由于逐句解释运行程序，所以速度比编译型的慢。但是较慢的运行速度仅明显表现在 M 文件初次运行时。因为 M 文件一经运行便将其代码存放在内存中，再次运行该文件时，MATLAB 将直接从内存中取出代码运行，大大加快了运行速度。

从功能上讲，M 文件大大扩展了 MATLAB 的能力。通过工具箱，MATLAB 才被应用到控制、信号处理、小波分析、系统辨识、图像处理、优化、样条分析、神经网络和金融财政等各个方面。而这些工具箱全部是由 M 文件构成的。从这点上来讲，如果不了解 M 文件，就不算彻底了解 MATLAB。

由于 M 文件是解释性的程序语言，且以复数矩阵为基本运算单位，所以 M 文件无论从形式、结构还是语法规则等方面都比一般的计算机语言简单、易写、易读。另外，MATLAB 本身是用 C 语言写的，M 文件的语法又与 C 语言十分相像，因此熟悉 C 语言的用户可以轻松地掌握 MATLAB 的编程技巧。

2. M文件的形式

M 文件有命令文件和函数文件两种形式。这两种文件的扩展名相同，都是.m。

当用户要运行的命令较多时，直接从键盘上逐行输入命令比较麻烦，利用命令文件就

可以轻松地解决这一问题。用户可以将一组相关命令编辑在同一个 ASCII 码命令文件中。运行时只需输入文件名字，MATLAB 就会自动按顺序执行文件中的命令。

函数文件是另一种形式的 M 文件，它的第一句可执行语句是以 function 引导的定义语句。在函数文件中的变量都是局部变量。

- 命令文件

利用命令文件中的语句可以访问 MATLAB 工作区(Workspace)中的所有数据。运行过程中，产生的所有变量均是全局变量。除非用户运用 clear 命令将它们清除，否则这些变量一旦生成，就一直保存在内存空间中。

运行一个命令文件相当于从 Command Window 窗口中按顺序连续运行文件里的命令。由于命令文件只是一串命令的集合，因此程序不需要预先定义，而只需按 Command Window 窗口的命令输入顺序将命令编辑在命令文件中就可以了。注意，不要忘记文件的扩展名是“.m”。接下来将用具体例子说明如何通过命令文件画出下列分段函数所表示的曲面。

$$p(x_1,x_2)=\begin{cases}0.5457e^{-0.75x_2^2-3.75x_1^2-1.5x_1} & x_1+x_2>1\\ 0.7575e^{-x_2^2-6x_1^2} & -1<x_1+x_2\le 1\\ 0.5457e^{-0.75x_2^2-3.75x_1^2+1.5x_1} & x_1+x_2\le -1\end{cases}$$

(1) 打开 MATLAB 的 M 文件编辑器，编写以下内容：

```
%first.m    This is my first example.
a=2;b=2;
clf;
x=-a:0.2:a;y=-b:0.2:b;
for i=1:length(y)
   for j=1:length(x)
      if x(j)+y(i)>1
         z(i,j)=0.5457*exp(-0.75*y(i)^2-3.75*x(j)^2-1.5*x(j));
      elseif x(j)+y(i)<=-1
         z(i,j)=0.5457*exp(-0.75*y(i)^2-3.75*x(j)^2+1.5*x(j));
      else z(i,j)=0.7575*exp(-y(i)^2-6.*x(j)^2);
      end
   end
end
axis([-a,a,-b,b,min(min(z)),max(max(z))]);
colormap(flipud(winter));surf(x,y,z);
```

(2) 选择 File | Save 命令，将所有文件保存在磁盘中，并命名为 first.m。

(3) 在 Command Window 窗口中输入文件名，运行结束后可看到如图 3.1 所示的图形。

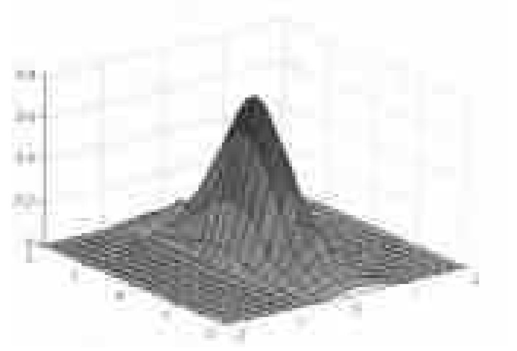


图 3.1 一个分段函数所表示的曲面

说明：

- ◆ 符号“%”引导的是注释行，不予执行。
- ◆ 不需要用 end 作为 M 文件结束的标志。
- ◆ 若用户把文件 first.m 存放在自己的工作目录(假如名为 e:\work)上，那么在运行 first.m 之前，应该先使 e:\work 处于 MATLAB 的搜索路径上。最简单的方法是在 MATLA 的 Command Window 窗口中先运行 cd e:\work，或将其加到 MATLAB 的搜索路径中去。

- 函数文件

如果 M 文件的第一行包含 function，此文件就是函数文件。每一个函数文件都定义一个函数。事实上，MATLAB 提供的函数命令大部分都是由函数文件定义的，这说明函数文件非常重要。从使用的角度来看，函数是一个“黑箱”，把一些数据送进并经加工处理，再把结果送出来。从形式上看，函数文件区别于命令文件之处是：命令文件的变量在文件执行完后保存在内存中，而函数文件内定义的变量仅在函数文件内部起作用。执行完函数文件后，这些内部变量将被清除。接下来说明如何通过 M 函数文件画出上例分段函数的曲面。

(1) 打开 MATLAB 的 M 文件编辑器，编写以下内容：

```
function  y= vectoraverage( a)
% 向量元素的平均值。
% 语法：vectoraverage(a)，其中 a 为输入向量。
% 当输入非向量时，给出错误信息。
[m,n]=size(a):
if(~((m==1)|n==1))|(m==1&n==1))  %判断输入是否为向量。
error('Input must be a vector')  %给出错误信息。
end
y=sum(a)/length(a);
```

(2) 保存文件 vectoraverage.m，该文件定义了名为 vectoraverage 的新函数，此函数的用法与别的函数一样。

(3) 在 Command Window 窗口中运行以下命令，可以求得 1～100 的平均值。

```
vectoraverage(1:100)
ans=
 50.5000
```

对以上内容说明如下：

- ◆ 执行第一行命令是为了指明该文件是函数文件，并定义函数名、输入参数和输出参数。函数名可以是 MATLAB 中任何合法的字符。输入和输出的参数根据实际需要设置，参数类型可以是数值，也可以是字符串。在本例中，输入参数是向量 a，输出参数是 y，为数值型。

  变量 a 对函数文件来讲是局部的。当该函数被调用结束后，变量 a 不再存在(如果变量 a 在程序运行前就已经存在的话，程序运行后它不会受到影响)。这一点可以用 who 命令检证。

  在 M 文件前面，连续几行带符号“%”的注释行有两个作用：一是随 M 文件

全部显示或打印时，直接起解释提示作用；二是供 help 命令联机查询使用。

◆ 在 MATLAB 命令窗中运行以下 help 命令，可得到帮助信息。

```
help vectoraverage
%向量元素的平均值.
语法：vectoraverage(a)，其中 a 为输入向量.
当输入非向量时，给出错误信息向量元素的平均值。
```

◆ 利用 lookfor 命令对关键词进行搜索，获取帮助信息。

```
lookfor vectoraverage
…(省略部分输出)
vectoraverage.m %向量元素的平均值
…(省略部分输出)
```

说明：

◆ 运行 help 命令后，所显示的是 M 文件注释语句中的第一个连续块。至于与第一连续块被空行所隔离的其他注释语句，将被 MATLAB 联机帮助系统忽略。

◆ 该例 lookfor 命令运行后，显示出 vectoraverage 函数文件的第一行注释。一般来说，为了利用 MATLAB 对关键词的搜索功能，用户在编制文件时，应在第一行注释中尽可能多地包含该函数的特征信息。

◆ 为了使 lookfor 命令能对用户目录上的 M 文件的关键词进行搜索，必须使用户在 MATLAB 的搜索路径上。

3.2　数据类型和全局变量

1.　数据类型

MATLAB 提供 6 种基本的数据类型，由于 MATLAB 主要用于数学处理，并且是以数组、矩阵运算为基础，因此这 6 种数据类型可以是一维、二维或多维。

这 6 种数据类型是：双精度型(double)、字符型(char)、稀疏型(sparse)、8 位型(unit8)、细胞型(cell)和结构型(struct)。一般把这 6 种类型的二维变量称为矩阵，其中最常用的是双精度矩阵和字符型数组。因为在 MATLAB 中，都采用双精度计算，而 MATLAB 所提供的绝大多数函数都是对双精度矩阵和字符串进行操作，而其他的几种数据类型多用于一些特殊的场合。稀疏型用于对稀疏矩阵的操作，细胞型和结构型变量则多用于大型的软件中，这与 C 语言的结构等数据类型是类似的，8 位型数据类型用于对图像的处理。

变量的数据类型可以通过调用函数 isa 来查看，调用格式为：

```
isa(变量名,数据类型)
```

数据类型即 double、char、sparse 等关键字，需要用“''”括起来，如同一个字符串一样。如果在函数 isa 所列的数据类型和变量的数据类型一致，返回值为 1。否则，返回值为 0。例如：

```
%双精度型。
```

```
a=[1,2;3,4]
a-
1    2
3    4
isa(a,'double')
ans=
    1
b=1+2*I
b =
   1.0000 + 2.0000i
isa(b,'double')
ans =
     1
isa(b,'char')
ans =
     0
%字符型数组。
'Matlab'
%双精度型稀疏矩阵，实际只存储了矩阵中的非零元素。
Speye(8)
%细胞数组，数组中的每个元素可为不同类型的不同维数。
{17  'Matlab' speye(8)}
%结构数组，相当于数据库中的记录，把相关的数据列在一起，这些数据称为这个记录的属性，不同属性的数据类型可以不同。
s.name='matlab';
s.time='15days';
s.number=magic(5);
s
s =
      name: 'matlab'
      time: '15days'
      number: [5x5 double]
%8 位型为无符号整数，最大可以表示 255，即一个 Byte，不能进行属性运算，常在图像处理中表示灰度等级。
unit8(3)
```

从以上的例子可以看出，数组是所有数据类型的基础，数组又可以分为字符型、结构型、细胞型和数值型，数值型又可以分为双精度型和 8 位型，稀疏型实际上也可算是双精度型的一种，因为稀疏类型中的每个元素都是以双精度形式存储和计算的。

MATLAB 所有的数据类型都支持一定的函数和运算方法，子一层的数据类型支持其父一层的所有运算，例如双精度型数据支持所有数组一层的运算。稀疏性变量支持双精度型的所有运算。表 3.1 列出了 MATLAB 中所有的数据类型及其支持方法。

表 3.1 数据类型及其支持的方法

数据类型	支持的方法
数组	多维下标、组合、转置、行列初等变换、数组变形、求维数、个维的大小
字符型	字符函数计算，计算时自动转换出双精度型
结构型	属性引用

续表

数据类型	支持的方法
细胞型	各元素用{}引用
数值型	Find函数、复数函数、冒号算符
双精度型	数学算符、逻辑算符、矩阵函数、数学函数
稀疏型	稀疏函数和所有双精度型的计算
8位型	存储特性

在 MATLAB 中，用户可以为已有的数据类型增加新的使用方法，甚至可以定义新的数据类型，并与MATLAB中已有的数据类型一样使用。

2. 全局变量

全局变量用命令global定义，如运行命令global X Y Z，就会将X，Y，Z定义为全局变量。

正如前面所说的，函数文件的内部变量是局部的，与其他函数文件及MATLAB内存相互隔离。但是，如果在若干函数中，把某一变量定义为全局变量，那么这些函数将公用这个变量。全局变量的作用域是整个MATLAB的工作区，即全程有效，所有的函数都可以对它们进行存取和修改。因此，定义全局变量是函数之间传递数据的一个手段。

值得指出的是：在程序设计中，全局变量虽然可带来某些方便，但却破坏了函数对变量的封装，降低了程序的可读性和可靠性。因而，在结构化程序设计中，全局变量是不受欢迎的。尤其当设计程序较大，子函数较多时，全局变量将给程序调试和维护带来许多不便，所以一般不提倡使用全局变量。如果一定要用全局变量，最好给它起一个特别的名字，以避免和其他变量混淆。

3.3 程序结构

从理论上讲，只要有顺序、循环和分支3种基本程序结构，MATLAB就可以构成任何一种程序并完成相应的工作。与大多数计算机语言一样，MATLAB有设计程序所必须的程序结构，即顺序结构、循环结构及分支结构。在MATLAB语言中，循环由while和for语句实现，分支结构由if语句实现。

MATLAB虽然不像C语言那样具有丰富的控制结构，但MATLAB自身的强大功能弥补了这个不足，使用户在编程时几乎感觉不到困难。 MATLAB语言是一种完善易用的高水平矩阵编程语言。

1. 顺序结构

MATLAB的顺序结构实际上就是复合表达式构成的语句。复合表达式由分号或逗号分隔的几个表达式构成。当表达式后面接分号时，表达式的计算结果虽不显示，但中间结果仍保留在内存中。若程序是命令文件，则程序运行完后，中间变量都予以保留；若程序是

函数文件，则运行完程序后，中间变量将被全部删除。

2. 循环结构

在实际生活中，常常会遇到许多有规律的重复运算，因此在程序中就需要将某些语句重复执行。一组被重复执行的语句称为循环体。它每循环一次，都必须做出是继续重复或是停止的判断，这个判断所依据的条件称为循环的终止条件。MATLAB 语言提供了两种循环方式：for.end 循环和 while.end 循环。

● for.end 循环

这种循环允许一组命令以固定的和预定的次数重复。循环的一般形式是：

```
for x=array
{commands}
end
```

在 for 和 end 语句之间的{commands}按数组中的每一列执行一次。

举例如下：

```
% 一个简单的 for 循环示例。
for i=1:10;                          %i 依次取 1,2,…,10。
y(i)=i;                      %对每个 i 值，重复执行由该命令构成的循环体。
end;
y                                    %要求显示运行后数组 y 的值。
y =
     1     2     3     4     5     6     7     8     9    10
```

说明：

◆ for 循环不能用 for 循环内重新赋值循环变量 i 的方法来终止。

◆ 语句 1：10 是一个标准的 MATLAB 数组创建语句。在 for 循环内接受任何有效的 MATLAB 数组。

◆ 有一个等效的数组方法来解给定的问题时，应避免用 for 循环。例如，上面的第一个例子可被重写为：

```
i=1:10;
y=i
y =
     1     2     3     4     5     6     7     8     9    10
```

两种方法得出同样的结果，而后者执行更快、更直观，要输入的内容较少。

◆ 为了得到最大的速度，在 for 循环(while 循环)被执行之前，应预先分配数组。例如，前面所考虑的第一种情况，在 for 循环内每执行一次命令，向量 y 的维数增加 1。这样就使得 MATLAB 每通过一次循环对 y 分配更多的内存，这当然要花费一定的时间。为了可以不执行这个步骤，for 循环的例子应重写为：

```
y=zeros(1,10)    %预先为 y 分配内存。
for i=1:10;
y(i)=i;
end;
```

现在，只有y(i)的值需要改变。

◆ for循环可按需要嵌套。

● while.end循环

for循环以固定次数来求一组命令的值。相反，while循环以不定的次数求一组语句的值。循环的一般形式是：

```
while expression
{commands}
end
```

只要在表达式里的所有元素为真，就执行while和end语句之间的{commands}。通常，表达式的求值给出一个标量值，但数组值也同样有效。在数组情况下，所得到数组的所有元素必须都为真。

Fibonacci数组的元素满足Fibonacci规则：$a_{k+2}=a_k+a_{k+1}$, $(k=1,2,\ldots)$；且$a_1=a_2=1$。现要求该数组中第一个大于10000的元素。

```
a(1)=1;a(2)=1;i=2;
while  a(i)<=10000
      a(i+1)=a(i-1)+a(i);    %当现有的元素仍小于10000时，求解下一个元素。
      i=i+1;
end;
i,a(i),
i =
    21
ans =
  10946
```

说明： 在这个例子里，以i=2开始。只要a(i)<=10000为真(非零)，就一直求while循环内的命令值。由于a(i)不断增大，以致于总会出现a(i)>10000，即a(i)<=10000是假(零)，于是while循环结束。

3. 分支结构

很多情况下，命令的序列必须根据关系的检验有条件地执行。在编程语言里，这种逻辑由某种if.else.end结构来提供。最简单的if.else.end结构是：

```
if expression
 {commands}
end
```

如果在表达式中的所有元素为真(非零)，那么就执行if和end语言之间的{commands}。例如：

```
% 一个简单的分支结构。
cost=10;number=12;
if number>8
   sums=number*0.95*cost;
end,sums
sums =
  114.0000
```

假如有两个选择，结构将是：

```
if 表达式
表达式为真时的命令序列。
else
表达式为假时的命令序列。
end
```

在这里，如果表达式为真，则执行第一组命令；如果表达式为假，则执行第 2 组命令。当有 3 个或更多的选择时，if.else.end 结构则采用以下形式：

```
if 表达式 1
表达式 1 为真时的命令序列。
elseif 表达式 2
表达式 2 为真时的命令序列。
elseif 表达式 3
表达式 3 为真时的命令序列。
elseif
…
else
所有的表达式均不为真时的命令序列。
end
```

现在知道了如何用 if.else.end 结构来控制循环，就有可能提出一种合理的方法来跳出或中断 for 循环和 while 循环。

下例为求 Fibonacci 数组的中第一个大于 10000 的元素另一种方法。

```
a(1)=1;a(2)=1;
for i=2:50
  a(i+1)=a(i.1)+a(i);
  i=i+1;
if a(i)>10000 %当现有的元素大于 10000 时，跳出 for 循环。
break
end
end
i,a(i),
i =
     21
ans =
  10946
```

结果与以前的例子相同。

说明： 这个例子演示了估算的另一种方法。在这种情况下，for 循环要执行足够多的次数。if.else.end 结构检验要看是否 a(i)>10000，如果是，break 命令强迫 for 循环提早结束。如果一个 break 语句出现在一个嵌套的 for 循环或 while 循环结构里，那么 MATLAB 只跳出 break 所在的那个循环，不跳出整个嵌套结构。

3.4 程序流控制

本节将系统介绍有关程序流控制的命令echo、input、pause、break和keyboard。

1. echo命令

通常，执行M文件时，文件的命令不会显示在Command Window窗口中。用echo命令可以使文件命令在执行时可见，这对程序的调试和演示极为有用。对应于命令文件和函数文件，echo的作用稍微有些不同，如表3.2所示。

表 3.2 echo命令的使用

命令格式	用　途	备　注
echo on	显示其后的所有被执行命令文件的命令	仅用于命令文件
echo off	不显示其后的所有被执行命令文件的命令	
echo	在上面两种状态中切换	
echo FileName on	使FileName指定文件的命令在执行中被显示出来	适用于命令文件和函数文件
echo FileName off	终止显示FileName文件的执行过程	
echo on all	文件的执行过程是否被显示的切换开头	
echo off all	使其后所有被执行文件的过程不被显示	

注意： 当把echo运用于某一函数文件时，该文件将不被编译执行，而是被解释执行。这样，函数文件在执行过程中，每一行都可被观察到。由于这种解释执行不大有效，因而仅用于程序调试。

2. input命令

命令input提示用户从键盘输入数值、字符串或表达式，并接受该输入。下面是几种常用的格式：

```
a=input('please input a number:')
```

运行该命令后，将给出如下文字提示，并等待键盘输入：

```
Please input a number:
```

用户可以输入数字或表达式，也可以输入字符串(两端必须有单引号)，按Enter键确认后，该输入被赋给变量a。

```
a=input('please input a string:','s')
```

运行该命令后，也给出提示：

```
Please input a string:
```

等待用户输入。在此提示后输入的任何内容(不管是数字还是字符)一律被当作字符串

赋给变量 a。

注意： 当输入为一个字符串时，可以在输入中使用转义符，这样，“/”之后的字符将和“/”一起，构成新的含义(例如：/n 代表按 Enter 键)。如果要输入“/”，则要使用“//”。事实上，这与大多数程序设计语言是相同的。

3. pause命令

pause 命令使程序运行暂停，等待用户按任意键继续。pause 命令在程序调试以及需要看中间结果时特别有用。pause 的用法有两种，pausc 表示暂停执行程序；pause(n)表示在继续执行前，暂停 n 秒。

4. keyboard命令

keyboard 命令与 input 一样重要。在程序遇到 keyboard 命令时，MATLAB 将会暂停程序的运行，并且调用机器的键盘命令进行处理。处理完自己的工作之后，输入，然后按下 Enter 键，程序将继续运行。M 文件中有了它之后，就可以在程序调试或执行程序时修改变量。

5. break命令

语句导致包含 break 命令的最内层 while、for、if 语句终止。通过使用 break 语句，可以不必等循环的自然结束，而是根据循环内部另设的条件，决定是否退出循环，是否结束 if 语句。在很多情况下，这是必须的。

6. error

使用函数 error 可以显示错误信息。函数 error 的语法格式为：

```
error('error string')
```

如果在 M 文件中调用函数 error，则显示在单引号中的字符串，并且停止执行程序。假如下面的 M 文件代码包含在 myfile.m 中：

```
if n<1
error('n must be 1 or greater.')
end
```

当 n 小于 1 时，显示的错误信息如下：

```
???Error using==>myfile
n must be 1 or greater.
```

7. warning

在 MATLAB 中，警告信息与错误信息类似。当执行程序中，显示警告信息时，程序不停止运行。显示警告信息可以用函数 warning。例如：

```
warning('warning string')
```

函数 lastwarn 用于显示最后发生的警告信息，与函数 lasterr 用法类似。例如：

```
l=lastwarn; %将最后的警告信息赋值给字符串 l。
warning(lastwarn) %显示最后的警告信息。
```

8. 外部系统命令

在 MATLAB 环境中，可以发出 Windows 或 Dos 系统命令，“!”命令能起到这种作用。它使得 MATLAB 将后面的命令传到相应的操作系统。这个过程通常称作使用外部系统命令。MATLAB 有 4 种形式的外部系统命令，就根据命令形式的尾部参数可以区分开来。

- 同步实时处理命令
  如果在外部系统命令(“!”) 之后没有其他附加的参数，那么 MATLAB 会打开一个新的窗口作为该命令的运行窗口。MATLAB 系统要等到该命令完成之后，才开始接受新的命令。例如，!dir 将在其中列出当前的内容。MATLAB 执行完上述命令后，便回到了 MATLAB 的提示符状态，等待新的命令。
- 后台处理命令
  如果外部命令行以字符“&”结束，MATLAB 则将此命令作为后台命令处理，不必等到完成该命令后，才接受和执行新的 MATLAB 命令。在需要打开新的应用程序时，可以使用该命令。例如，发出命令!notepad&后，将启动 Notepad(记事本)作为一个新的 Windows 任务。类似地，也可以发出 DOS 命令，这时将打开一个 DOS 窗口，但作为后台处理命令，此时可以在 MATLAB 命令窗口中执行其他的命令。
- 图标后台处理命令
  第 3 种外部命令形式是以字符“|”结尾。这个命令的作用与后台处理命令相同，只是用一个图标作为后台命令打开的窗口。这个命令可以用在对命令的运行结果不感兴趣的情况下，例如 DOS 的批处理命令等。
- MATLAB 的 DOS 命令
  另一种使用操作系统命令的方法是使用 MATLAB 的 DOS 命令。MATLAB 系统将部分 DOS 命令作为自己的命令，执行时将这些命令直接传递给 DOS 操作系统。运行的结果在 Command Window 窗口中显示。有时，也打开一个 DOS 窗口，但该窗口在执行下一条新命令时自动关闭。也可以在命令后面加上“&”和“|”来改变窗口的形态。

3.5 函数调用和参数传递

MATLAB 中的函数调用及参数传递比较复杂。但是这两件工作又是编写高质量的 M 文件所不可少的。一个较大的计算任务可以分成若干个较小的任务。这意味着，一个程序可以由若干个函数组成，并通过函数调用来实现控制转移和相互之间的数据传递。

1. 函数调用

在 MATLAB 中，调用函数的常用形式是：

[输出参数 1，输出参数 2,…]=函数名(输入参数 1，输入参数 2,…)

注意： 函数调用时各参数出现的顺序，应该与函数定义时的顺序一样，否则就会出错。

函数调用可以嵌套，一个函数可以调用别的函数，甚至调用它自己(递归调用)。

举例如下：

给定两个实数 a、b，一个正整数 n，给出 k=1,2,…,n 时的所有(a+b)^n 和(a.b)^n(本例限定不大于 10)。

(1) 建立函数文件。

```
function[out1,out2]=power(a,b,n)
%power.m 计算(a+b)^n 和(a-b)^n。
out1=(a+b)^n;
out2=(a-b)^n;
```

(2) 建立调用上述函数文件的命令 example1.m。

```
a=input('Please input a=:');
b= input('Please input b=:');
addpow=zeros(1:10);
subpow=zeros(1:10);
for k=1:10
  [addpow(k),subpow(k)]=power9a,b,k);
end
addpow
subpow
```

说明： 在本例中，命令文件 example1.m 对函数 power 每做一次调用，将传入 3 个参数 a、b 和 k，送出两个结果 addpow 和 subpow。

用递归调用形式计算 n 的阶乘。

(1) 编写递归调用函数文件 factor.m。

```
function f=factor(n)
%factor.m 计算 n 的阶乘。
if n==1
   f=1;
return;
         else
f=n*factor(n.1);
return;
         end
```

(2) 运行函数文件。

```
factor(6)
ans=
     720
```

说明： 递归函数是一类算法编程的最直接方式，在程序设计中有重要地位。递归调用可使程序简洁易读。

2.　参数传递

MATLAB在函数调用上有一个与众不同之处，函数所传递的参数具有可调性。凭借这种特性，一个函数就可以完成多种功能。

传递参数数目的可调性来源于以下两个MATLAB永久变量：

```
nargin   %函数体内的给出调用该函数时的输入参数数目。
nargout  %函数体内的给出调用该函数时的输出参数数目。
```

只要在函数文件中包含这两个变量，就可以准确地知道该函数文件被调用时的输入参数和输出参数的数目，从而决定如何处理。

举例如下：

```
function sa = circle(r,s)
%CIRCLE      plot a circle of radii r in the line specified by s.
%r           指定半径的数值。
%s           指定线色的字符串。
%sa          圆面积。
%
% circle(r)              利用蓝实线画半径为r的圆周线。
% circle(r,s)            利用串 s 指定的线色画半径为r的圆周线。
% sa=circle(r)           计算圆面积，并画半径为r的蓝色圆面。
% sa=circle(r,s)         计算圆面积，并画半径为r的s色圆面。
if nargin>2
   error('输入参数太多。'); %如果输入参数数目大于2，给出错误信息，程序结束。
end;
if nargin==1
%如果输入参数数目为1，则该输入参数指定半径的数值，默认圆周线线色为蓝色。
   s='b';
end;
clf;
t=0:pi/100:2*pi;
x=r*exp(i*t);
if nargout==0      %如果输出参数数目为0，则画圆周线。
   plot(x,s);
else               %如果输出参数数目大于0，则画圆面，并计算圆面积输出给sa。
   sa=pi*r*r;
   fill(real(x),imag(x),s)
end
axis('square')
```

3.6　MATLAB的数据接口

MATLAB具有灵活、方便的数据输入输出功能。因为有关的内容比较庞杂，本节只作简要介绍。

1. 数据输入

用户可以用多种方式向 MATLAB 系统输入数据，并根据数据格式选择最合适的输入方式。

- 显式的输入

  针对数据量比较小的情形，可以在 MATLAB 的 Command Window 窗口中，用键盘直接输入数据。这种方法就是前面介绍的矩阵直接输入法，即用方括号将矩阵元素按行的顺序括起来，每行用分号隔开，行内各元素用空格或逗号分开。

- M 文件形式的输入

  如果数据量较大，且不是以计算机可读形式存在时，最有效的办法是用某种文本编辑器直接编写一个包含数据矩阵的 M 文件。最后通过执行 M 文件达到数据输入的目的。例如编辑一个数据 M 文件 data.m：

```
%数据 M 文件
a=1;
b=2;
c=3;
A=[1,2,3;4,5,6;7,8,9];
%将 data.m 文件存入当前工作路径目录中，然后直接调用这个 M 文件，就可以将其中的数据
加载到现在的工作空间中。
data
who
Your variables are:
A  a  b  c
A
A =
     1     2     3
     4     5     6
     7     8     9
a
a=
1
```

- ASCII 码数据文件的输入

  利用 MATLAB 可以直接读入 ASCII 码的数据文件。ASCII 码数据文件中的数据形式必须是一个矩阵，要求数据文件每一行的数据个数必须相同，每行数据对应于矩阵的每一行，每行的元素用空格分开。用户可以将数据编辑成一个 ASCII 码文件，或者由其他的程序将所输入的数据写到一个 ASCII 码文件中，ACSII 码文件中的数据由 MATLAB 的命令 load 装入，命令形式是：

```
load 文件名(扩展名)
```

  该语句在 MATLAB 工作区中创建一个与文件名(无文件扩展名)相同的变量，该变量表示的矩阵即是 ASCII 码文件的数据组成的矩阵。

- 低层 I/0 输入方式

  MATLAB 提供了文件低层操作函数。可以直接打开文件(fopen)和读文件(fread)，

这种方法主要用于装入某种特定格式的数据文件，这种数据文件可能是其他的应用程序生成和创建的。

- MEX 动态程序输入

  如果已经有一些子程序(如 C 子程序或 FORTRAN 子程序)可以用来读取某些特定格式的数据文件，那么可以开发 MATLAB 的动态链接 MEX 子程序，与已有的子程序链接在一起，将数据文件转换成 MATLAB 的 MAT 数据文件，再用 load 函数将有关的数据载入 MATLAB 系统中。

- 外部程序转换

  如果已有的数据文件格式比较复杂。用户可以开发 FORTRAN 或 C 程序，将数据文件直接转换成 MATALB 的 MAT 数据文件，再用 load 命令装入到 MATLAB 系统中。在这种情况下，用户须对 MAT 文件结构有较深入的了解。

2.　数据输出

MATLAB 系统中输出数据的方式主要有以下 5 种：

- 小型矩阵输出

  对于数据较小的数据矩阵，通过输出的设置，可以用 diary 命令生成 Command Window 窗口部分内容的副本文件。该文件是文本文件，可以直接用普通文本编辑软件修改。它记录了 Command Window 窗口的命令行，以及 MATLAB 的屏幕输出内容。这样可以将 diary 文件贴接到其他的文件或报告中。例如：

```
diary dataout.m
A=zeros(5)
A =
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
B=ones(5);
diary off
who
Your variables are:
A  B
```

用 type 命令将其显示出来。

```
type dataout.m
A=zeros(5)
A =
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
B=ones(5);
diary off
```

- ASCII 码数据输出

  利用 ASCII 选项的 save 命令，可以生成一个 ASCII 码的数据文件。利用这种方法用户可以存储某些指定的变量。例如下列语句：

```
A=rand(4,3)
save temp.dat A.ascii
```

  将生成名字为 temp.dat 的 ASCII 码文件，包含矩阵 A 的全部数据，如：

```
0.9501    0.8913    0.8214
0.2311    0.7621    0.4447
0.6068    0.4565    0.6154
0.4860    0.0185    0.7919
```

- 低层 I/O 输出

  利用 MATLAB 的低层 I/O 函数 fopen 和 fwrite，或者其他低层 I/O 函数，可将数据写入某个特定格式的文件中。这种方式主要用于将输出的数据写成某些其他应用程序所需要的数据格式。
- MEX 程序输出

  如果已经有某些子程序可以将数据写成特定格式，那么可以开发 MEX 子程序直接调用那些子程序，把最后形成的数据格式作为某些应用程序的输入数据。
- MAT 格式输出

  利用 MATLAB 的 save 命令，可以将数据存储成 MAT 格式文件，再开发一些 C 或 FORTMN 程序，将 MAT 文件转换成某些特殊格式。例如，将 MATLAB 的图像数据存入到 MAT 文件中，然后利用 C 或 FORTRAN 程序将 MAT 的图像数据转换成一些标准的图像格式文件，如 PCX、TIF 和 GIF 格式等。

3.7 文件的 I/O 操作

访问一个文件之前，必须先打开这个文件，在读写操作完成之后再关闭这个文件。每个被打开的文件对应一个代号，MATLAB 正是通过对应这个文件的代号来操作这个文件。

MATLAB 中的文件读写函数如表 3.3 所示。

表 3.3 MATLAB文件读写函数

函数名	功能
fclose	关闭文件
feof	测试文件结束
ferror	查询文件 I/O 的错误状态
fgetl	读文件的行，忽略按 Enter 键换行符
fgets	读文件的行，包括按 Enter 键换行符
fopen	打开文件
fprintf	把格式化数据写到文件或屏幕上

续表

函 数 名	功 能
fread	从文件中读二进制数据
frewind	返回到文件开始
fscanf	从文件中读格式化数据
fseed	设置文件位置指示器
ftell	获取文件位置指示器
fwrite	把二进制数据写到文件里

下面介绍其中几个常用的文件 I/O 函数。

- fopen()：打开文件或获得打开文件信息，其语法格式为：

  f_id＝fopen(文件名，'允许模式')：以'允许模式'指定的模式打开文件名所指定的文件，返回文件标识：f_id；'允许模式'可以是下列几个字符串之一：

  - 'r'　打开文件进行读操作(默认形式)。
  - 'r+'　打开文件进行读和写操作。
  - 'w'　删除已存在文件中的内容或生成一个新文件，打开进行写操作。
  - 'w+'　删除已存在文件中的内容或生成一个新文件，打开进行读和写操作。
  - 'W'　非自动刷新写操作，用于磁带驱动器。
  - 'a'　生成并打开一个新文件或打开一个已存在的文件，进行写操作，在文件末尾追加。
  - 'a+'　生成并打开一个新文件或打开一个已存在的文件，进行读或写操作，在文件末尾追加。
  - 'A'　非自动刷新追加操作，用于磁带驱动器。

  在上述字符串中加上一个字符“t”，如"at"，用来区分文本文件和二进制文件，强制使用文本模式打开文件。在上述字符串中加上一个字符 b，则强制使用二进制模式打开文件(此为默认模式)。

  [f_id,message]=fopen(文件名，'允许模式'，格式)：其打开文件方式如上面的形式，同时返回文件标识和打开文件信息两个参数，另外用格式指定数据格式。用来定义文件中数据格式的字符串，可以允许不同格式机器间的文件共享。允许格式字符串有：

```
'native'或'n'
    'ieee.ie'或'l'
    'ieee.be'或'b'
    'vaxd'或'd'
    'vaxg'或'g'
    'cray'或'c'
    'ieee.le.l64'或'a'
    'ieee.be.l64'或's'
```

`f_id=fopen('all')` %返回一行向量，包括所有打开文件标识，但不包括 0，1，2，向量的元素个数与打开文件数相同。

`[f_name,f_type,f_format]=fopen(f_id)` %返回指定文件的全称文件名字符串 f_name，允许模式字符串 f_type，格式字符串 f_format。不合法的 f_id 返回空字符串。

如果 fopen 成功打开文件，则返回文件标识 f_id，message 内容为空。如果不能成功打开，则返回 f_id 值为-1，message 中返回一个有助于判断错误类型的字符串。

有 3 个 f_id 值是预先定义的，不能打开或关闭：

- ◆ 0 表示标准输入，一直处于打开读入状态。
- ◆ 1 表示标准输出，一直处于打开追加状态。
- ◆ 2 表示标准错误，一直处于打开追加状态。

- fclose()：关闭一个或多个已打开的文件，其语法格式为：

```
status=fclose(f_id)   %关闭指定文件，返回 0 表示成功，返回-1 表示失败。
status=fclose('all')  %关闭所有文件，返回 0 表示成功，返回-1 表示失败。
```

- fread()：从文件中读二进制数据，其语法格式为。

[A,count]=fread(f_id,size,'精度') %从指定文件中读入二进制数据，将数据写入到矩阵 A 中；可选输出 count 返回成功读入元素个数；f_id 为整数文件标识，其值由 fopen() 函数得到；可选参数 size 确定读入多少数据，如果不指定参数 size，则一直读到文件结束为止。参数 size 合法选择有以下几种：

- ◆ n　　读入 n 个元素到一个列向量。
- ◆ inf　　读到文件结束，返回一个与文件数据元素相同的列向量。
- ◆ [m,n]　　读入足够元素填充一个 m×n 阶矩阵，填充按列顺序进行，如果文件不够大，则填充 0。

'精度'表示读入数据精度的字符串，控制读入每个值的数据位，这些位可以是整数型、浮点值或字符。

[A,count]=fead(f_id,size,'精度',skip) %可选参数 skip，指定每次读操作跳过字节数。如果'精度'是某一种位格式，则每次读操作将跳过相应位数。

MATLAB 中的'精度'如表 3.4 所示，表中还给出了相应 C 和 Fortran 语言的精度格式。

表 3.4 MATLAB中的精度字符串

MATLAB	C和Fortran	说　明
'char'	'char*1'	字符格式：8 位
'schar'	'signed char'	带符号位的字符：8 位
'uchar'	'unsigned char'	无符号字符：8 位
'int8'	'integer*1'	整数：8 位
'int16'	'integer*2'	整数：16 位
'int32'	'integer*4'	整数：32 位
'int64'	'integer*8'	整数：64 位
'uint8'	'integer*1'	无符号整数：8 位
'uint16'	'integer*2'	无符号整数：16 位
'uint32'	'integer*4'	无符号整数：32 位
'uint64'	'integer*8'	无符号整数：64 位
'float32'	'real*4'	浮点数：32 位

续表

MATLAB	C 和 Fortran	说　明
'float64'	'real*8'	浮点数：64 位
'short'	'short'	整数：16 位
'int'	'int'	整数：32 位
'long'	'long'	整数：32 位或 64 位
'ushort'	'unsiged short'	无符号整数：16 位
'uint'	'unsiged int'	无符号整数：32 位
'ulong'	'unsigned long'	无符号整数：32 位或 64 位
'float'	'float'	浮点数：32 位
'double'	'double'	浮点数：64 位
'bitN'		带符号整数：N 位(1≤N≤64)
'ubitN'		无符号整数：N 位(1≤N≤64)

- fwrite()：向文件中写入二进制数据，其语法格式为：

```
count=fwrite(f_id,A,'精度')    %将矩阵 A 中的元素写入指定文件，将其值转换为指定的精度。
count=fwrite(f_id,A,'精度',skip)    %可用参数 skip 指定每次写操作跳过指定字节。
```

- fscanf()：从文件读入格式化数据，其语法格式有：

```
A=fscanf(f_id,'格式')   %从由 f_id 所指定的文件中读入所有数据，并根据格式串进行转换，并返回给矩阵 A，格式字符串指定被读入数据的格式。
[A,count]=fscanf(f_id,'格式','size')    %读入由指定数量的数据，并根据格式字符串进行转换，并返回给矩阵 A，同时返回成功读入的数据数量 count；参数 size 合法有以下几种：
```

◆ n　　　　读入 n 个元素到一个列向量。
◆ inf　　　读到文件结束，返回一个与文件数据元素相同的列向量。
◆ [m，n]　读入足够元素填充一个 m×n 阶矩阵，填充按列顺序进行，n 可以取 inf，m 不能取 inf。

'格式'字符串由普通字符和转换规格组成：转换规格指匹配数据类型，包括字符“%”，可选宽度，转换字符。如：%−8.6e，其中“%”表示开始字符，“−”表示带正负号，“8.6”表示数字域宽度和精度，“e”表示转换字符。在“%”与转换字符之间可以出现一个或多个如下字符：

“*”　　表示跳过匹配的数值。
数字　　表示数值宽度。
字母　　表示接收对象的大小：如“h”表示短，%hd 表示短整数，“l”表示长，%ld 表示长整数，%lg 表示双精度浮点数。

在'格式'字符串中合法的转换字符包括以下几种：

%c　　表示字符序列，数量由域宽度指定。
%d　　表示十进制数。

%e，%E，%f，%F，%g，%G　表示浮点数。
%i　表示带正负号整数。
%o　表示八进制整数。
%s　表示一系列非空白字符。
%u　表示带正负号十进制整数。
%x，%X　表示带正负号十六进制整数。
[…]　表示一系列字符。

- fprintf()：向文件中写入格式化数据，其语法格式为：

count=fprintf(f_id，'格式'，A，…)　%将矩阵A或其他矩阵的实部数据以'格式'字符串指定的形式进行规格化，并将其写入指定的文件中，其返回值为写入数据的数量。

fprintf('格式'，A，…)　%将A或其他参数的值以格式给定的形式输出到标准输出，并显示在屏幕上。格式字符串同fscanf()中的格式字符串一样。

- fgets()：以字符串形式返回文件中的下一行内容，包含行结束符。其语法格式为：

str=fgets(f_id)　%返回文件标识为f_id的文件中的下一行内容，如果遇到文件结尾(EOF)，则返回-1。所返回的字符串中包括文本行结束符，用fgetl()则返回的字符串中不包括行结束符。

str=fgets(f_id,n)　%返回下行中最多n个字符，在遇到行结束符或文件结束(EOF)时不追加字符。

- fgetl()：以字符串形式返回文件中的下一行内容，但不含行结束符，其语法格式为：

str=fgetl(f_id)　%返回文件标识为f_id的文件中的下一行内容，如果遇到文件结尾，则返回-1。所返回的字符串中不包括行结束符，用fgets()可以包括行结束符。

- ferror()：查询MATLAB关于文件输入、输出操作的错误，其语法格式有：

message=ferror(f_id)　%将标识为f_id的已打开文件的错误信息返回给message变量。

Messase=ferror(f_id，'clear')　%返回标识为f_id的已打开文件错误信息并清除指定文件错误指示器。

[message，errornum]=ferror(…)　%返回指定文件错误信息，并返回最近文件输入、输出操作错误状态数errornum；如果指定文件最近输入、输出操作成功，则message值为空，errornum值为0。

- feof()：测试文件结尾(EOF)，其语法格式为：

eoftest=feof(f_id)　%测试指定文件是否设置了EOF；如果返回1，则表示设置了EOF指示器，返回0表示未设置。

- fseek()：设置文件位置指示器，其语法格式为：

status=fseek(f_id，offset，原始值)　%重新设置指定文件中位置指示器，从原始值相对移动指定的offset字节。

offset>0　表示位置指示器向文件结尾移动offset个字节。
offset=0　表示不改变位置指示器。
offset<0　表示向文件开头移动位置指示器|offset|个字节。
原始值合法值有以下几个：

'bof' -1： 表示文件开始位置。

cof or 0： 表示文件中当前位置。

'eof'or 1： 表示文件结尾(*OF)。

返回值 status 为 0 表示移动操作成功，为-1 表示操作失败。

- ftell()：获取文件指示器的位置，其语法格式为：

```
position=ftell(f_id)  %返回指定文件指示器的位置，position 为非负整数，表示从文件开头到当前位置的字节数。返回-1 表示查询失败，用 ferror() 可获取更多的错误信息。
```

- frewind()：倒回到文件开头，其语法格式为：

```
frewind(f_id)   %设置指定文件的位置指示器到文件开头。对于磁带驱动器中的文件，该函数不起作用，该函数不产生错误信息。
```

3.8 M 文件的调试

从程序设计的角度来讲，MATLAB 语言比其他的程序设计语言在说明结构上都简单。但是 MATLAB 有自己严密的语法，用户必须按语法的要求编写 MATLAB 程序，否则同样会产生一些错误。

许多的语法错误都可以在程序执行之前查出，这样的错误也容易处理。由于 MATLAB 是运行解释环境，没有独立的编译过程，所以在程序运行过程中，很可能会碰到在编程时不易发现的错误，而且很难按照运行的过程去追踪出错的地方。原因之一是 MATLAB 函数调用对 MATLAB 的工作区是局部的和封闭的，即使要求 MATLAB 输出中间的计算结果，也很难发现是在什么地方出现错误。

MATLAB 提供了 M 文件的调试功能，可以对 M 文件函数进行调试。MATLAB 的调试功能帮助用户确定 MATLAB 程序代码中的错误，在函数运行期间的任何时刻都可以调试查看 MATLAB 工作区的变量值，查看函数调用的栈管理，以及逐行地运行 M 文件。

调试为用户提供了命令行交互式接口，您可以通过 Command Window 窗口的菜单进行操作。

1. 调试的主要功能

调试主要是分析 MATLAB 程序中的隐含错误，可以发现和更正以下两类错误。

- 语法错误

  这类错误主要包括函数名拼写和括号遗漏等错误。当然，这类错误在程序运行时，MATLAB 系统自己可以检测到，并且会指出 M 文件中可能出错的行号。

- 运行错误

  这类错误通常是算法错误，而在语法上是正确的。这类错误会导致不正确的计算结果，而 MATLAB 系统不会发现出错的地方，这就必须使用调试。

一般来说，运行错误往往难于跟踪。因为当被调用的函数由于出错而退出函数体时，函数局部工作区的变量就全部消失了，这时可以用下列技巧发现某些运行错误：

- ◆ 在 M 文件中，将某些语句后的分号去掉，迫使 M 文件输出一些中间计算结

果，以便发现可能的算法错误。

◆ 在 M 文件中，加入 keyboard 语句。keyboard 语句可以设置程序的断点。
◆ 将函数 M 文件改为子程序 M 文件，这样可以在 MATLAB 工作区查看中间计算结果。

最后一种方法是使用调试，调试比其他技巧优越的地方在于它可以深入到函数的工作区，可以设置或清除断点、按行执行程序等。

2. **调试命令**

MATLAB 的调试命令如表 3.5 所示。

表 3.5 MATLAB中的调试命令

命令名称	命令功能
dbstop	设置程序断点
dbclear/dbclear all	清除程序(所有)断点
dbcont	恢复程序运行至程序结束或到另一个断点
dbdown/dbstep in	深入下层局部工作区
dbstack	列函数调用关系
dbstatus	列出所有断点
dbstep	指定执行步数
dbtype	列出行号内的 M 文件
dbup	向上层改变局部工作区
dbquit	退出调试状态

3. **调试的使用**

如果要检测函数 M 文件是否有某些错误，可以对该函数进行调试。在函数中设置断点，帮助分析和发现程序的错误原因。在调试时，若函数子程序遇到断点，则将子程序暂时停下来，并将断点处的命令行显示出来，同时显示键盘输入的提示符。用户可以在该提示符之后输入任何合法的 MATLAB 命令。

使用 MATLAB 的调试命令时，应该记住以下几点：

- 调试命令只能对函数 M 文件使用，不能对其他的 M 文件使用。
- M 文件的断点与编译过的 M 文件相关联，因此如果 M 文件被清除或是被重新编辑，那么所有的断点即被取消。

4. **调试器的使用**

MATLAB 提供了一个基于 GUI 界面的调试。使用它，可以对函数进行调试。接下来通过一个具体帮助用户掌握它的一些基本功能。

本例将对于任意随机向量，画出鲜明标志该随机向量均值、标准差的频数直方图，如图 3.2 所示，或给出绘制这种图形的数据。

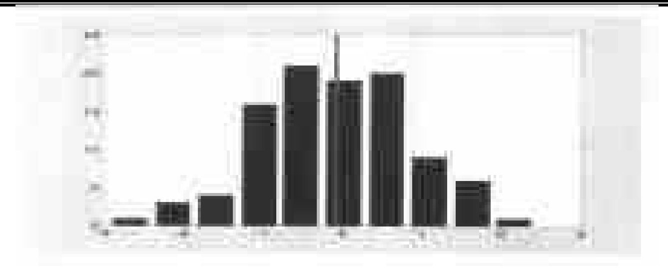


图 3.2 带均值、标准差标志的频数直方图

(1) 根据题目要求写出以下两个 M 文件。

```
[histzzy.m]
function [nn,xx,xmu,xstd]=histzzy(x)
xmu=mean(x);
xstd=std(x);
[nn,xx]=hist(x);
if nargout==0
  barzzy(nn,xx,xmu,xstd)
end
[barzzy.m]
function barzzy(nn,xx,xmu,xstd)
clf,
bar(xx,nn);hold on
Ylimit=get(gca,'YLim');
yy=0:Ylimit(2);
xxmu=xmu*size(yy);
xxL=xxmu/xmu*(xmu.xstd);
xxR=xxmu/xmu*(xmu+xstd);
plot(xxmu,yy,'r','Linewidth',3)        %<9>
plot(xxL,yy,'rx','MarkerSize',8)
plot(xxR,yy,'rx','MarkerSize',8),hold off
```

(2) 初次运行以下命令后，得到运行出错的提示。

```
randn('seed',1),x=randn(1,100);histzzy(x);
??? Error using ==> plot
Vectors must be the same lengths.
Error in ==> E:\matlab6.1\work\barzzy.m
On line 9  ==> plot(xxmu,yy,'r','Linewidth',3)          %<9>
Error in ==> E:\matlab6.1\work\histzzy.m
On line 6  ==>   barzzy(nn,xx,xmu,xstd)
```

生成的图形如图 3.3 所示。

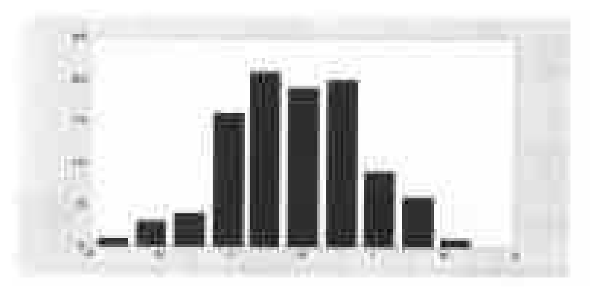


图 3.3 运行出错时所得到的不完整图形

单击命令窗口中有下划线的错误提示，系统将弹出对应的M文件，光标出现在错误行处。

(3) 初步分析错误原因。

初步分析错误原因是命令 plot(xxmu,yy,'r','Linewidth',3)中的向量 xxmu 和向量 yy 的维数不一致。因为 xxmu 和 yy 是内部变量，所以为了求得向量 xxmu 和向量 yy，必须设置断点，进入函数内部进行调试。

(4) 设置断点。

根据运行出错的提示，在函数 histzzy()的第6行和函数 barzzy()的第9行设置两个断点。

(5) 在命令窗中运行以下命令，进入动态调试 randn('seed',1),x=randn (1,100);histzzy(x); 运行暂停在第一个断点处，如图3.4所示。

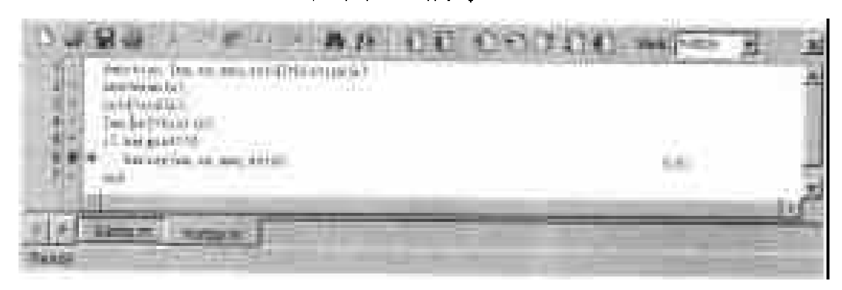


图 3.4 运行暂停在第一个断点处

(6) 深入被调文件内部。

单击工具栏中的按钮进入 barzzy()函数，如图3.5所示。

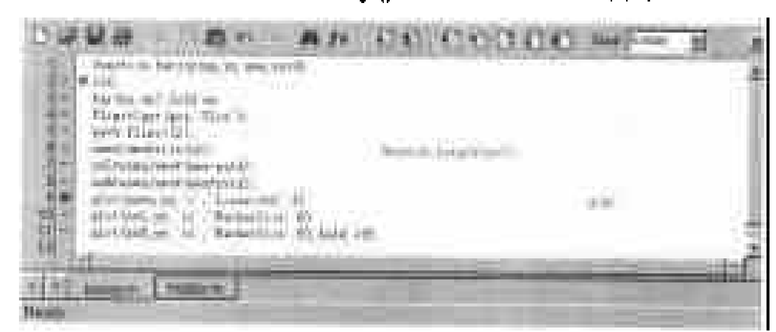


图 3.5 动态调试深入barzzy.m

(7) 连续执行，直到另一个断点。将鼠标指针移至向量 yy 处，可以查看它的值，如图3.6所示。
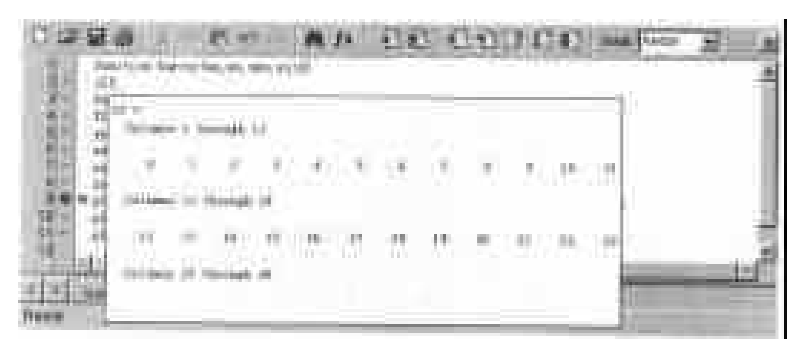


图 3.6 变量值的鼠标指针观察法

也可以查看 xxmu 的值为：

```
xxmu=
   .0.0959    .2.4934
```

显然 xxmu 的值不正确，它应该和 yy 的维数一致。修改 xxmu 的求解表达式如下：

```
xxmu=xmu* ones(1,length(yy));
```

(8) 修改程序，停止第一轮调试，重新运行。
停止第一轮调试，清除断点，在命令窗中重新运行以下命令：

```
randn('seed',1),x=randn(1,100);histzzy(x);
```

程序运行通过，结果如图 3.6 所示。

第 4 章　MATLAB 6 图形绘制基础

数据的可视化在实际工作和科学试验中具有重要作用，MATLAB 作为一个强大的数学工具，有丰富的数据可视功能。它可以给出原始数据的二维、三维甚至四维的图像，同时可以对生成的图像进行各种修饰与控制。此外，MATLAB 还可以绘制符号函数的图形。

MATLAB 提供了高级图形命令和低级图形命令(即句柄图形)，本章只介绍 MATLAB 的高级图形命令，关于句柄图形的内容将在第 9 章详细介绍。

4.1　创建 MATLAB 二维图形

二维图形是绘制复杂图形的基础，绘制复杂图形所用的许多概念其实都是二维图形的扩展，所以本节首先介绍二维图形，并简单介绍一些特殊的二维图形的绘制方法和范围应用。

4.1.1　创建简单的二维图形

MATLAB 具有强大全面的绘图功能，是数据可视化的有效工具。 图形仍以矩阵为数据单位，并且主要通过描点法绘图。现在举两个简单的例子，介绍如何用 plot()命令绘制简单图形。

首先看一个离散数据和离散函数的可视化例子。

%用图形表示离散函数 $y=|(n-6)|^{-1}$。

```
n=7:12;                                  %产生一组自变量数据。
y=1./abs(n-6);                           %计算相应点的函数值。
plot(n,y,'r*','MarkerSize',20)           %用红花标出数据点。
grid on                                  %画坐标方格。
```

运行结果如图 4.1 所示。

接下来来看一个简单连续的正弦函数可视化的例子，由于 MATLAB 是基于描点法作图，所以必须把自变量离散化。假设只作 x 在[0,2π]区间的图像，先在[0,2π]内取 100 个点，这一步可以用 x=linspace(0, 2*pi, 100)实现。现在在 Command Window 窗口中执行下列语句：

```
x=linspace(0,2*pi, 100);    %生成离散自变量，100 个点的 x 座标。
```

```
y=sin(x);            %函数定义对应的 y 座标。
plot(x,y);           %用 plot()函数作图。
```

运行结果如图 4.2 所示。

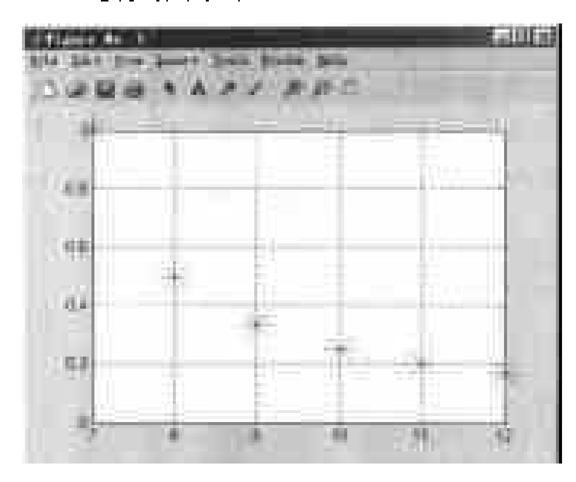


图 4.1　离散函数的可视化

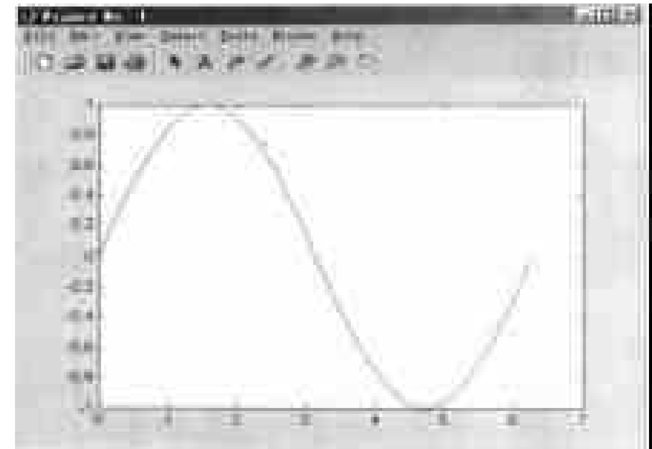


图 4.2　连续正弦函数可视化

命令中的 plot(x,y)是一个基本的线性刻度绘图函数，x 代表横坐标，y 代表轴坐标。

4.1.2　修饰简单的二维图形

图 4.1 和图 4.2 虽已显示出函数的曲线，但是在实际应用中，还希望加上坐标轴的说明、图名、显示栅格线、标记点等操作，这都可以用相应的函数来完成。接下来介绍 MATLAB 中的一些基本的曲线修饰函数。

1.　改变颜色与线形，增加图例及文本信息

继续使用上面的例子，现在要改变曲线的颜色，给函数加一个参数，即在 plot(x,y)命令中加'color')，其中 color 为颜色代码，例如 w 代表白色。MATLAB 中一些基本的颜色代码如表 4.1 所示。

表 4.1　基本颜色和线形

字　元	颜　色	字　元	图线型态
Y	黄色	.	点
k	黑色	O	圆
w	白色	x	X
b	蓝色	+	+
g	绿色	*	*
r	红色	-	实线
c	亮青色	:	:
M	锰紫色	-.	点虚线
		--	虚线

线形的用途在于：当在一个图形中有多条曲线时，可以用不同的型态点来区分它们。同样给 plot(x,y) 函数加一个参数，用 plot(x,y,'mark')，其中 mark 为 MATLAB 中已设定的一些特殊字符，例如输入命令“plot(x,y,'+')”，即表示所绘制的曲线由“+”构成。

给曲线加图例也是区分和标志曲线的方法，其中 legend()函数可以实现此功能，它用文本确认数据集。

为使图形的含义更加明确，有时需要在图中加入特定的文本说明，text 命令就可以实现此功能。Text 的基本命令格式为 text(x,y,'string')，其中(x,y)是文本字符串的起始坐标，string 是文本内容。

下面应用以上函数，在 Command Window 窗口中输入以下命令：

```
x=linspace(0, 2*pi, 100);
y=sin(x);
plot(x,y,'r', x,y,'*')  %先用红色画出正弦函数，再用“*”描点重画正弦函数。
legend('y=sin(x)')      %图例注解。
text(1.4,0.91,'波峰')   %图形注解。
```

运行结果如图 4.3 所示。

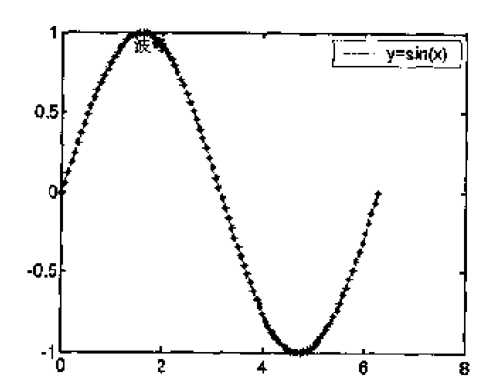


图 4.3 正弦函数曲线的颜色、线形和文本修饰

2. 坐标轴的说明与标记

仍使用上面的例子。MATLAB 用 xlabel()、ylabel()函数给坐标轴加以说明，用 grid 函数加栅格线。请看下面的例子，在 Command Window 窗口中继续执行如下命令：

```
xlabel('Input Value');        % x 轴注解。
ylabel('Function Value');     % y 轴注解。
title('一个正弦函数');        % 图形标题。
```

运行结果如图 4.4 所示。

xlabel()函数的一般语法形式为：

```
xlabel('string')或xlabel(…'PropertyName',PropertyValue,…)
```

string 是所加的具体注释，PropertyName 是字符串 string 的属性名，PropertyValue 是相应属性值。相应的 ylabel()函数和 zlabel()函数与此相同。

有关 grid 函数的具体例子请考看图 4.1。

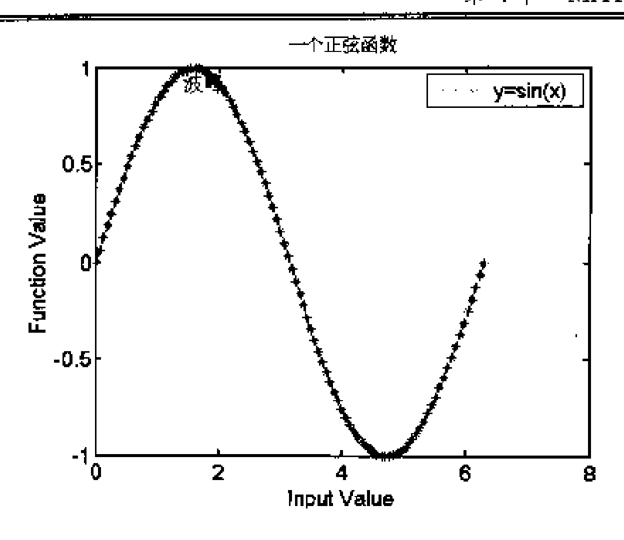


图 4.4 坐标轴的说明与标记

3. 定制图形坐标轴

在制作图形的过程中，有时会对坐标轴的相对尺度、长度及坐标系等有特定要求，例如想改变坐标轴的纵横比。在作三角函数、机械工程中的轮轴曲线时，要采用极坐标系等，MATLAB 中的 axis 函数就可以实现对坐标轴的控制。该命令一般可表示为：axis comm 或 axis('comm')，comm 是 MATLAB 内部提供的命令行，有很多。MATLAB 中常用的函数及说明，如表 4.2 所示。

表 4.2 坐标轴处理函数

函　数	说　明
axis([xmin xmax ymin ymax])	以 xmin xmax 设定横轴的下限及上限，以 ymin ymax 设定纵轴的下限及上限
axis auto	横轴及纵轴依照数据大小的上下限来订定，横轴及纵轴比例是 4:3
axis square	横轴及纵轴比例是 1:1
axis equal	将横轴纵轴的尺度比例设成相同值
axis xy	默认情况下使用卡氏座标，即将图原点设在左下角；横轴由左往右递增，纵轴由下往上递增
axis ij	使用矩阵格式，即将图原点设在左上角，横轴不变，纵轴由上往下递增
axis normal	以默认值画纵轴及横轴
axis off	将纵轴及横轴取消
axis on	恢复纵轴及横轴

注意： 上述各个函数的语法也可以将关键字放在括号内的单引号之间，如 axis('auto')。同时上述命令有一个辅助命令 v=axis，向量 v 记录了当前坐标轴的界限。

下面看一个用 axis 命令控制图形坐标轴例子。

在 Command Window 窗口中输入以下命令：

```
x=linspace(0,2*pi,30); y=sin(x); z=cos(x);
plot(x,y,x,z)
axis on
axis('square','equal')
```

运行结果如图 4.5 所示。

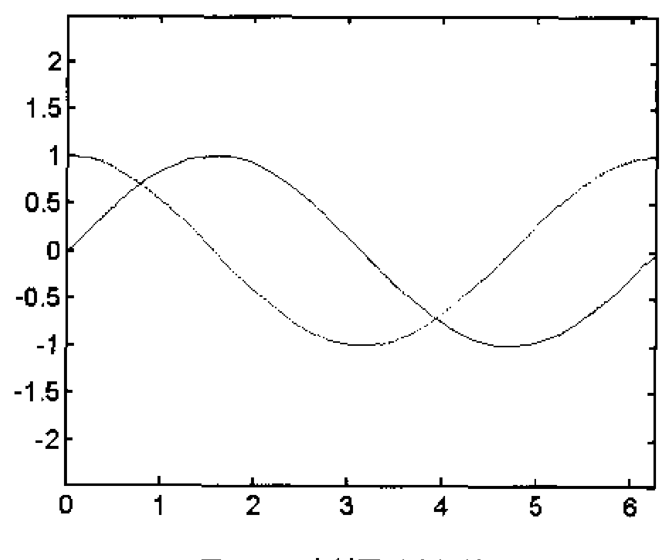


图 4.5 定制图形坐标轴

读者可以继续输入命令“v=axis”，看看有什么结果。

4.1.3 基本绘图函数

MATLAB 中的基本绘图函数有以下 4 种：

plot：x 轴和 y 轴均为线性刻度(Linear scale)。

loglog：x 轴和 y 轴均为对数刻度(Logarithmic scale)。

semilogx：x 轴为对数刻度，y 轴为线性刻度。

semilogy：x 轴为线性刻度，y 轴为对数刻度。

plot()的命令格式如下：

- plot(x,y,[linspec]) 其中 linspec 是可选的，用它来说明线型，这是 plot()函数的常用语法格式。
- plot(y) 以 y 的元素为纵坐标，相应元素下标为横坐标绘图。
- plot(x1,y1,x2,y2,...)使用该命令可以在一个图形窗口绘制多条曲线，具体内容在 4.1.4 节详述。

其他 3 个函数的用法用与 plot()相同，只是 X 轴或 Y 轴的刻度不同。

4.1.4 创建多个图形

作为预备知识，首先介绍图形窗口的概念图形窗口(figure window)，它是 MATLAB 中

用以显示图形的窗口，它与 Command Window 窗口是相互独立的，其属性由系统和 MATLAB 共同控制。在没有创建图形窗口之前，如果要绘制图形，MATLAB 将自动创建一个新的图形窗口。您也可以用 figure 函数直接创建新图形窗口，例如直接在 Command Window 窗口中输入“figure”，即创建了一个图形窗口，且为当前活动窗口；当 MATLAB 中已存在一个或多个图形窗口时，MATLAB 即默认最后一个窗口为当前活动窗口以输出图形。

注意： 在实际应用中，为说明问题，例如对比曲线参数变化对曲线形状的影响，经常要在一个图形中绘制多条曲线以方便对比，在 MATLAB 中有 4 种方法实现此目的。

1. 用plot()函数

用 plot()函数的第 3 种形式为：

```
plot(x1,y1,x2,y2,…)
```

即可在同一图形窗口绘出多条曲线。其中 x1,y1 为第一条曲线的坐标，x2,y2 为第 2 条曲线的坐标，依次类推，x1 可以与 x2 相等。请看下面的例子。

用图形表示连续调制波形 $y=\sin(t)\sin(9t)$ 及其包络线。

```
t=(0:pi/100:pi)';                   %长度为 101 的时间采样列向量。
y1=sin(t)*[1,-1];                   %包络线函数值，是 101×2 的矩阵。
y2=sin(t).*sin(9*t);                %长度为 101 的调制波列向量。
t3=pi*(0:9)/9;
y3=sin(t3).*sin(9*t3);
plot(t,y1,'r:',t,y2,'b',t3,y3,'bo')     %绘制 3 条曲线。
axis([0,pi,-1,1])                   %控制轴的范围。
```

该例绘出的曲线如图 4.6 所示。

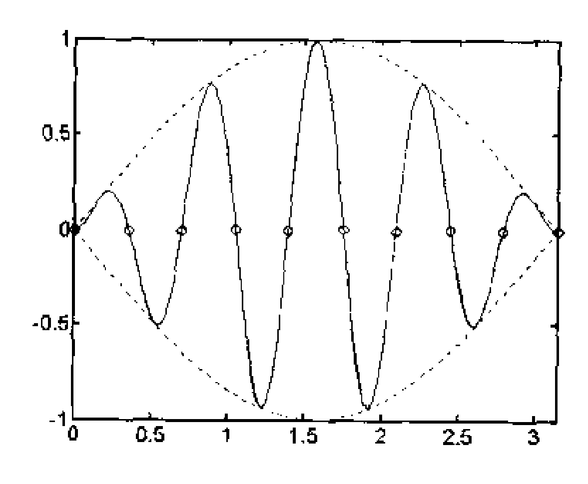


图 4.6　使用plot()函数绘制多条曲线

2. 用hold on命令

在已经存在图形的窗口内使用 hold on 命令后，再用新的 plot()命令， MATLAB 并不取消原有的图形，而是在已存在的图形中再添加新的图形，这样就在同一窗口中绘制了多

个图形，后来绘制的图形坐标如果与原坐标不同，系统将自动调整坐标轴。hold off 命令与 hold on 命令相反，使用该命令即释放当前窗口。

接下来在图 4.6 中的同一坐标下添加第 4 条曲线。

```
hold on
y4=-1:0.2:1;
plot(1.5,y4, '*')
```

该例绘出的图形如图 4.7 所示。

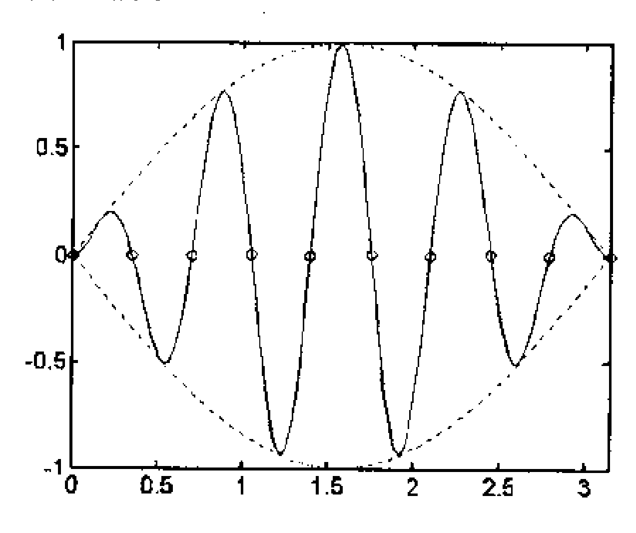


图 4.7 使用hold on命令绘制图形

3. 用subplot()函数

以上两种方法虽然可以在同一窗口中绘制多条曲线，但必须使用同一坐标系。显然，这在实际应用中有很大的局限性。如果要在同一窗口中用不同的坐标系绘出几组数据，就可以使用创建子图的方法。该方法的函数语法命令为 subplot(m,n,p)，该命令把当前图形窗口分成 m×n 个绘图区域，并选择第 p 个区域为当前活动区域。此时，只有该活动窗口响应绘图命令。如果需要改变激活的窗口，只改变 p 的值即可。下面是该方法的例子：

```
x=linspace(0,2*pi,30);
subplot(2,2,1); plot(x,sin(x));
subplot(2,2,2); plot(x,cos(x));
subplot(2,2,3); plot(x,sinh(x));
subplot(2,2,4); plot(x,cosh(x));
```

该例绘出的曲线如图 4.8 所示。

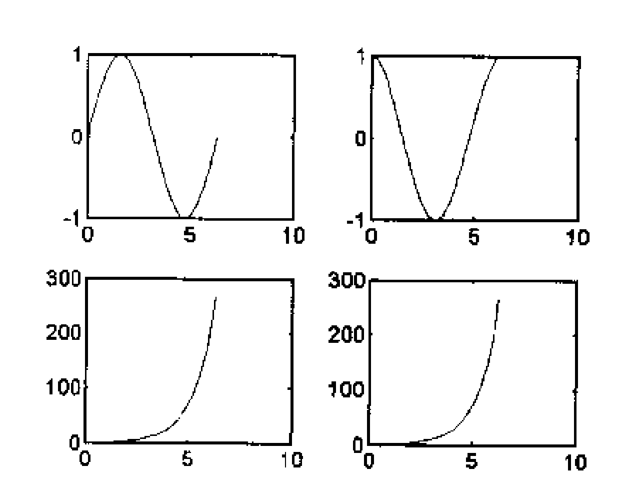


图 4.8 使用 subplot 函数绘制多条曲线

4. 创建多图形窗口

上面的方法都是在同一个窗口中绘图，此时该窗口中图形对象的某些属性有相同的取

值，有时在应用中不希望出现这种情况，而是需要把每个图形分别绘制在不同的窗口中以方便修改某一图形的属性，这时就需要使用创建图形窗口的命令——figure 命令。每执行一次没有参数的 figure 命令就创建一个图形窗口，相应地会生成该图形窗口的句柄以留给句柄函数调用。同时 figure 命令还可以将已有的窗口设为当前窗口，此时的操作就在当前窗口中进行。请看下面的例子。

```
x=linspace(0,2*pi,30);
figure
%生成第一个图形窗口，自动定义为一号窗口，如图 4.9 所示。
plot(x, sin(x));
xlabel('Input Value');       % x 轴注解
ylabel('Function Value');    % y 轴注解
title('一个正弦函数');         % 图形标题
legend('y=sin(x)')
```

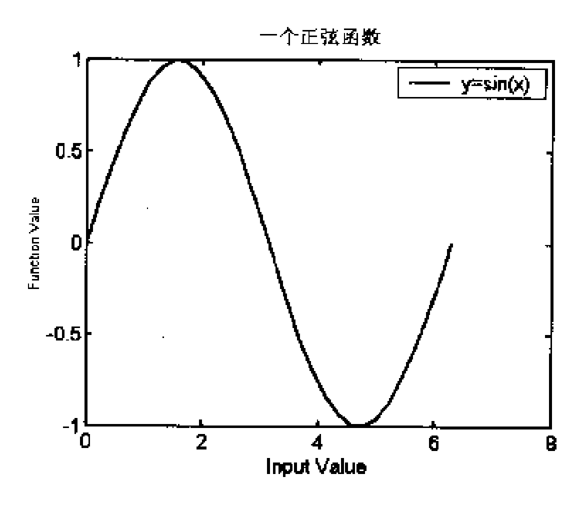


图 4.9　一号窗口

```
figure
%生成 2 号图形窗口，  自动定义为二号窗口，如图 4.10 所示。
```

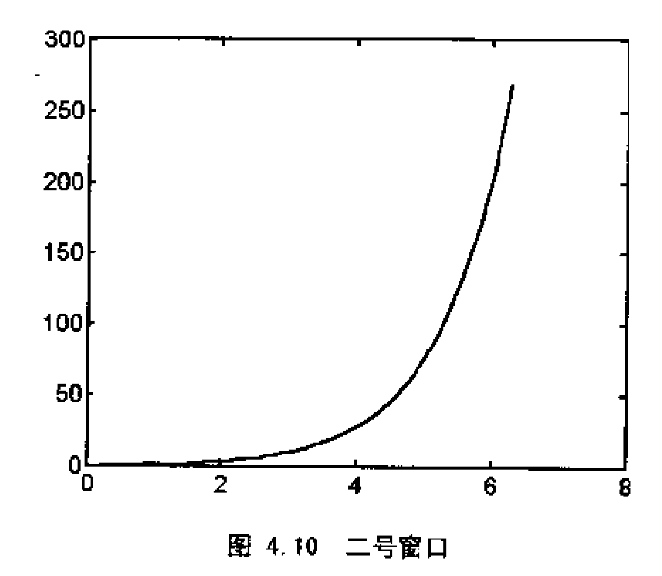


图 4.10　二号窗口

```
plot(x, sinh(x));
close %关闭当前窗口。
```

4.1.5 特殊的二维图形函数

MATLAB 中的特殊二维图形函数本节将对它们进行一一介绍，如表 4.3 所示。

表 4.3 特殊的二维图形函数

二维图形函数	说 明
bar	直方图(又称长条图)
errorbar	给图形加上误差范围
stem	柄图(又称针状图)
polar	极坐标图
hist	频数累计柱状图
rose	极坐标累计图
stairs	阶梯图
fplot	较精确的函数图形
fill	实心图
feather	羽状图
compass	矢量图(又称罗盘图)
quiver	向量场图(又称二维箭头图)

1. 直方图

在实际应用中，常会遇到离散数据，当需要比较数据、分析数据在总和中的比例时，直方图就是一种理想的选择，但要注意该方法适用于数据较少的情况。直方图的绘图函数有以下两种基本形式：

- bar(x,y) 绘制 m×n 矩阵的直方图。其中 y 为 m×n 矩阵或向量，x 必须单向递增。
- bar(y) 绘制 y 向量的直方图，x 向量默认为 x=1:m。

此外还可以给 bar()函数加 width、stacked、grouped 等参数，width 设置方条的宽度，默认值是 0.8，当 width>1 时，图形中的方条会互相覆盖；stacked 参数设置图形为堆积形式，此时直方图的方条数与矩阵的列数相同，相应每个方条画出矩阵的每行；grouped 参数设置直方图为组合形式。请看下面的例子：

```
close all; % 关闭所有的图形视窗。
x=1:10;
y=rand(size(x));
bar(x,y); % 绘制直方图
```

该例绘出的曲线如图 4.11 所示。

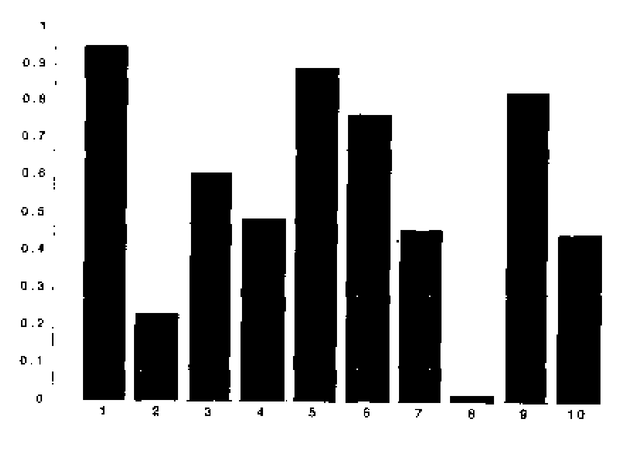


图 4.11 直方图示例

bar()函数还有 barh()和 errorbar()两种形式，barh()用来绘制水平方向的直方图，其参数与 bar()相同；当知道资料的误差值时，可用 errorbar()绘制出误差范围，其一般语法形式为：

```
errorbar(x,y,l,u)
```

其中 x,y 是其绘制曲线的坐标，l,u 是曲线误差的最小值和最大值，制图时，l 向量在曲线下方，u 向量在曲线上方。

或用 errorbar(x,y,e)绘制误差范围是[y-e,y+e] 的误差直方图。

下面看一个例子：

```
x=linspace(0,2,20)*pi;
y=sin(x)
e=std(y)*ones(size(x))； %设置误差为原始数据的标准差。
errorbar(x,y,e)； %绘制以标准差为误差范围的误差直方图。
```

该例绘出的曲线如图 4.12 所示。

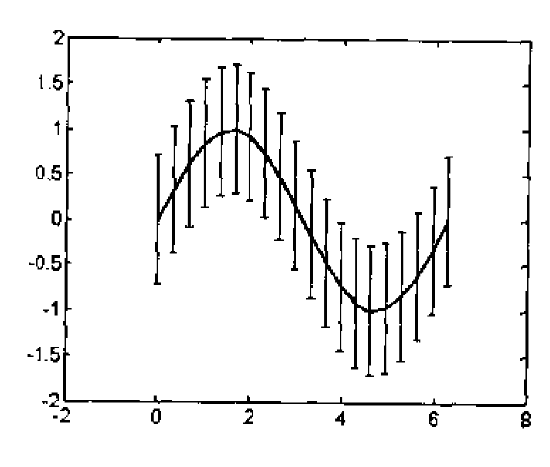


图 4.12 errorbar()函数的使用

2. 柄图

柄图又称火柴杆图或针状图，主要用来绘制数位信号。该图把每个数据点画成一条直线，在直线末端用点表示数据，所以形象地称作火柴杆图或针状图(大头针)。绘制此种图形的函数为 stem()函数，常用格式如下：

- stem(y) 向量 y 的值作为柄的长度从 x 轴延伸，x 值自动产生，当 y 为矩阵时，每一行的值在同一个柄上生成。
- stem(x,y) 绘制 x 对 y 的列向量的柄图，x 和 y 可以是同样大小的向量或矩阵。当 x 为行或列向量时，y 行数必须与 x 的长度相同。
- stem(...,'fill') fill 参数确定是否填充柄的头部。
- stem(...,Linespec) linespec 确定柄图线的属性，如线型、颜色及标记等。

下面是绘制柄图的一个简单的例子。

```
x=linspace(0,10,50);
y=sin(x).*exp(-x/3);
stem(x,y);        %绘制柄图。
```

该例绘出的图形如图 4.13 所示。

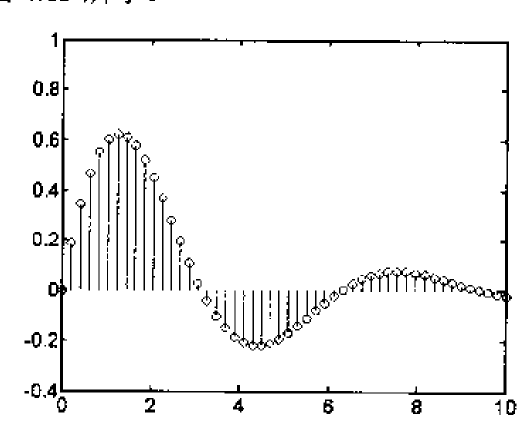


图 4.13 利用stem函数绘制柄图

3. 阶梯图

和柄图类似，stairs()函数也常用来绘制横坐标是时间序列的数位信号，又称阶梯图。不同的是 stairs()函数绘制出的阶梯图其相邻数据点间不用直线连接，而是相邻两点间的值全取起点数据的值。该函数的常用语法格式与 stem()函数类似的有：

```
stairs(Y)
stairs(X,Y)
stairs(…,Linespec)
```

变量的含义与 stem()函数类似。

stair()函数画出阶梯图例子如下所示：

```
x=linspace(0,10,50);
y=sin(x).*exp(-x/3);
stairs(x,y);          %绘制函数 y 的阶梯图。
title('stair 函数')
```

运行结果如图 4.14 所示。

4. 饼图

饼图与直方图的功能类似，都表示资料中某个分量在总量中所占的比例。它的基本命

令格式为：

- pie(x) 绘制向量 x 的饼图，x 中的值被 x/sum(x)规范化以确定饼图中每一片的大小。如果 sum(x)<=1，则直接用 x 中的值作为饼中片的大小；如果 sum(x)>1，则只画出饼图的一部分。
- pie(x,explode) 用来从 x 的饼图中去掉 explode 向量表示的片，explode 必须与 x 大小相同。explode 向量被置 1 的分量对应片与此饼图分开。
- pie(x,label) 用来标注饼图中片的名称。

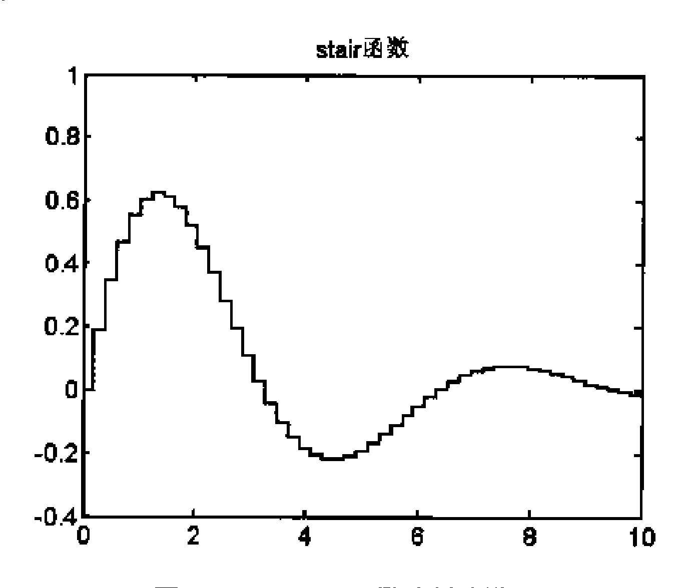


图 4.14 stairs函数绘制阶梯图

下面是一个用 pie()函数绘制饼图的例子。

```
x=[11.4,23.5,35.4,15.6];          %某工厂 4 个季度的生产量。
explode=zeros(size(x));
                                  %生成零向量。
[c,offset]=min(x);
                                  %c=1,求最小值的下标 offset ，c=1。
explode(offset)=1;                %指定占比例最小的一块和整个饼分离。
pie(x,explode);                   %绘制有分离的饼图。
```

运行以上程序，得到如图 4.15 所示的图形。

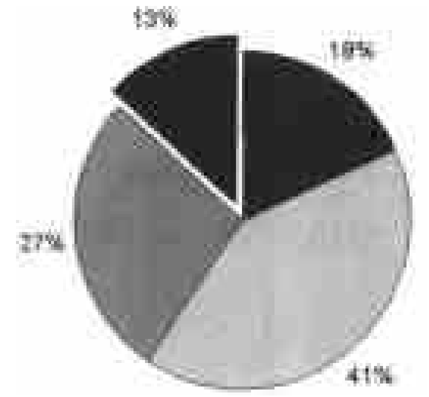


图 4.15 用pie()函数绘制的饼图

5. 频数累计柱状图

频数累计柱状图主要用于在笛卡尔坐标系中统计在一定范围内数据的频数，并用柱形图表示出来，以用大量的资料显示其分布情况和统计特性。函数 hist()的常用语法格式为：

- n=hist(y) 把 y 向量中的数据等划分为 10 个区间进行统计，最后画出 10 个柱形。如果 y 为矩阵，则按列计算。
- n=hist(y,x) 其中 y 为要统计的。当 x 为标量时，x 指定了统计的区间数；当 x 为向量时，以该向量中各元素为中心进行统计，区间数等于 x 向量的长度。
- n=hist(y,n) 其中 n 为要绘出的柱形数。

下面是柱形图的一个例子。

```
x=randn(5000, 1);          % 产生 5000 个 m=0, s=1 的高斯乱数。
hist(x,20);                % 20 代表长条的个数
```

该例绘出的图形如图 4.16 所示。

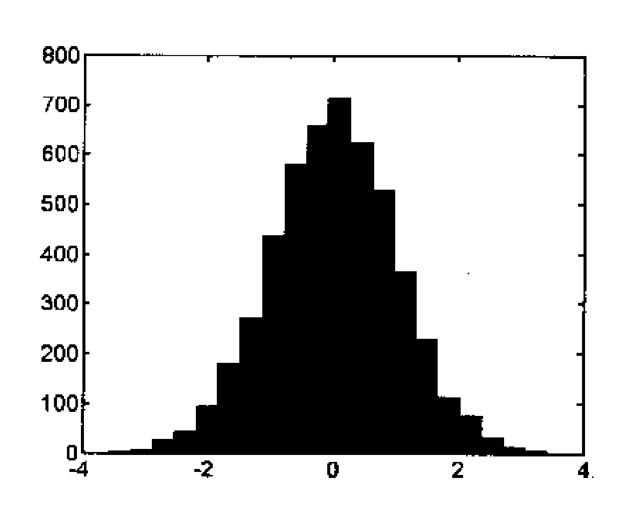


图 4.16 用hist()函数绘制的频数累计柱状图

6. 矢量图

函数 compass()用于显示起点在图形原点的矢量。该函数采用笛卡儿坐标系，并且在圆形栅格上进行图形的绘制。

下面举例来绘制显示 12h(小时)周期内的风向与风力的矢量图。首先定义两个矢量，分别为风向与风力。

```
wdir=[0 45 45 90 45 360 270 335 335 360 90];
wstr=[6 6 6 4 7 8 9 12 14 13 8];
```

将风向的角度转换为弧度，并绘制矢量图。代码如下：

```
rdir=wdir*pi/180;
[x,y]=pol2cart(rdir,wstr);
compass(x,y)
```

最后，在矢量图上添加一些标注：

```
desc={'风向和风力',
```

```
'北京气象台',
'8 月 11 日 12:00 到',
'8 月 12 日 15:00'}
dext(-22,12,desc)
```

上述代码执行的结果如图 4.17 所示。

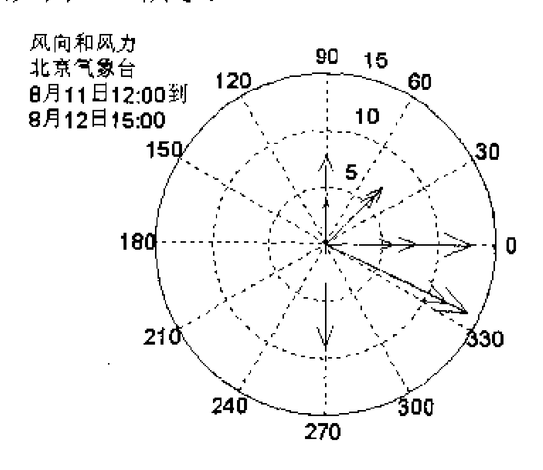


图 4.17　用compass()函数绘制的矢量图

7.　羽状图

feather 函数用于以箭头的形式显示一系列矢量，这些矢量的起点均匀地分布在一条与 x 轴平行的直线上。例如，创建以下两个矢量：

```
theta=90:-10:0;
r=ones(size(theta));
```

在创建羽状图之前，首先将矢量数据转换到笛卡儿坐标系下，同时将矢量 r 扩大 10 倍，以便箭头显示得更清楚。命令如下：

```
[u,v]=pol2cart(theta*pi/180,r*10);
feather(u,v)
axis equal
```

上述代码执行的结果如图 4.18 所示。

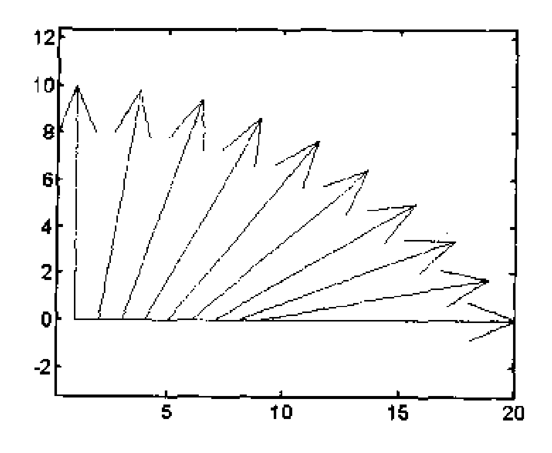


图 4.18　用feather()函数绘制的羽状图

绘制羽状图时，如果输入的参数是一个带复数的矩阵，假设为 z，则 MATLAB 将 z 的实部解释为矢量的 x 组元，而将 z 的虚部作为矢量的 y 组元。例如：

```
theta=linspace(0,2*pi,20);
z=cos(theta)+i*sin(theta);
feather(z);
```

代码执行的结果如图 4.19 所示。

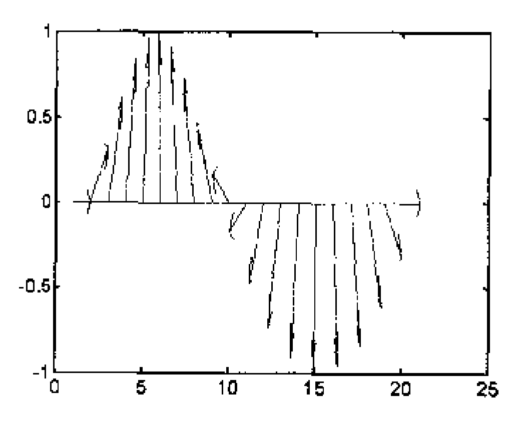


图 4.19 复数矩阵的羽状图

8. 极坐标图

极坐标图在工程计算中应用十分广泛。MATLAB 用 polar()函数绘制极坐标图，polar()函数的常用语法格式为：

- polar(theta,rho) 用角度 theta 对极半径 rho 作图。其中 theta 必须用弧度表示，如用角度需先转换。
- polar(theta,rho,s) theta 与 rho 同前，s 为曲线使用的线型。

应用如下，得到的结果如图 4.20 所示。

```
theta=linspace(0,2*pi);
r=cos(4*theta);
polar(theta, r);
title('极坐标图')
```

另外，还可以用 rose()函数在极坐标系中绘制频数累计柱状图——角度直方图(又称玫瑰图)。rose 和 hist 很接近，只不过是将数据大小视为角度，数据个数视为距离，并用极坐标绘制表示。该函数的常用语法格式为：

- rose(theta) 用相角 theta 绘制角度直方图。
- rose(theta,nbins) 其中 nbins 是一个整数，把 0~2π 分成等份，默认值为 20。
- rose(theta,x) 其中 x 是一个向量，用 theta 对向量 x 作图。

接下来绘制离散随机序列的角度直方图。

```
x=randn(1000,1);
rose(x);
title('随机序列的角度直方图');
```

运行后的结果如图 4.21 所示。

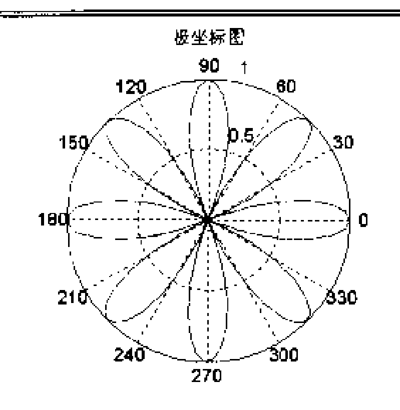


图 4.20 用polar()函数绘制羽状图

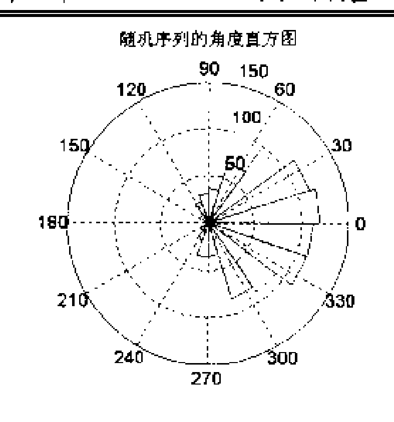


图 4.21 用rose()函数绘制的角度直方图

9. 精确绘图

用 plot()函数绘图时有时会失真，因此 MATLAB 又提供了一个用来精确绘图的函数 fplot()函数，这样可以对剧烈变化处进行较密集的取样。

fplot()函数的一般语法格式为：

- fplot(fun,lims) fun 给出要绘制的函数，可以是自建的 M 文件函数或以 x 为自变量的函数，其结果为相对向量 x 中各元素的行向量；lims 为 x 或 x 与 y 向量的变化范围，格式为 lims=[xmin,xmam]或[lims=[xmin,xmax,ymin,ymax]。
- fplot(fun,lims,tol) tol 参数指定绘图时相对误差的公差，其默认值是 2*e-3。
- fplot(fun,lims,n) 参数 n 确定绘制曲线的最大步长，当 n>=1 时，函数最少绘制 n+1 个点，绘图的最大步长限制在(1/n)*(xmax-xmin) 。
- fplot(fun,lims,linspec) linspec 指定绘图时所用线的属性。

该函数的应用如下，结果如图 4.22 所示。

```
fplot('sin(1./x)',[0.02 0.2]);        %sin(1/x)是一个在原点附近函数值变化很剧烈的函数
title('剧烈变化的函数');
```

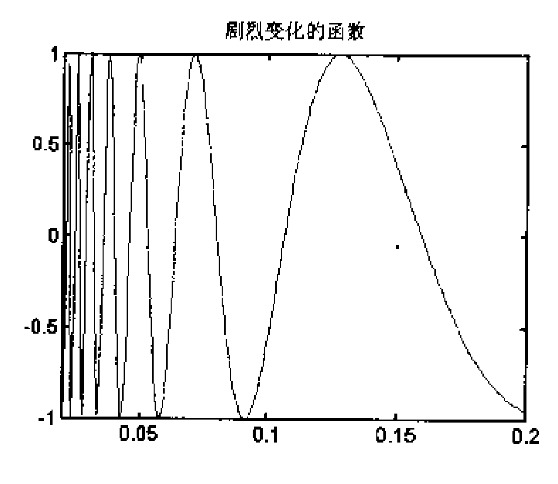


图 4.22 fplot()函数精确绘图

10. 向量场图

quiver()函数用于在二维平面的给定点绘制矢量，这里需要绘制由 x 组元和 y 组元组成

的矢量。向量场图通常是与 contour()函数(绘制等高线图)和 gradient()函数(求梯度)配合使用的。

该函数的应用可参见下例，结果如图 4.23 所示。

```
n=-2.0:.22:2.0;
[X,Y,Z]=peaks(n);
contour(X,Y,Z,10);            %创建轮廓图(等高线图)。
[U,V]=gradient(Z,.2);
quiver(X,Y,U,V)               %创建向量场图。
title('向量场图');
```

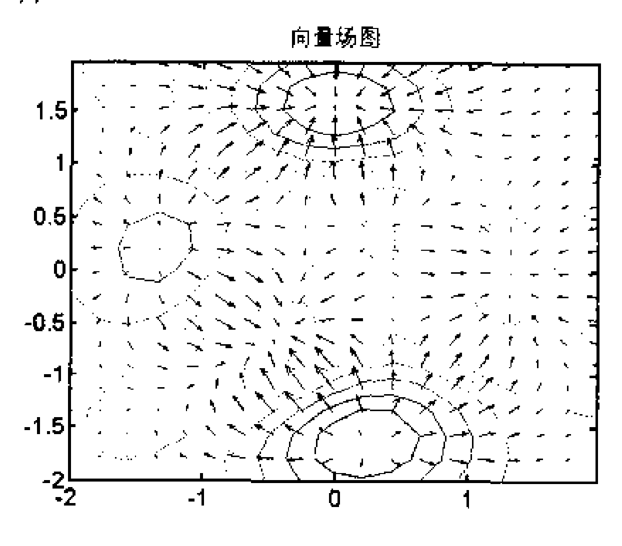


图 4.23 用quiver()函数创建的向量场图

4.2 创建 MATLAB 三维图形

工程中经常需要对三维数据进行处理，三维数据的分析要比二维数据的分析复杂得多，这时绘制三维原始数据或经过处理的数据的三维图形以辅助分析就更加重要了，本节主要介绍如何用 MATLAB 绘制能够满足各种需要的三维图形。

4.2.1 创建简单的三维图形

MATLAB 不仅提供了卓越的二维图形功能，其三维绘图功能也很强大。MATLAB 提供了绘制三维曲线和三维曲面的功能，包括一系列的图形修饰和色彩设置函数，例如设置图形的视角、光影效果、颜色配置、消隐和透视等，使绘制的图形更具有真实感和立体感。下面逐一介绍这些函数及功能。

创建三维图形的具体步骤如下：

(1) 准备绘图数据，这些数据可以是实际工作中采集的数据，也可以用各种命令或函数创建，以后的例子中常用 peaks()函数，即峰形函数来生成数据。

(2) 创建输出图形的窗口，例如用 figure()函数，此步骤可略去，而由绘图函数自动创建窗口。

(3) 调用三维绘图命令，例如 plot()、mesh()、surf()等函数，也可以是自己编写的绘图命令，根据需要绘制图形。

(4) 修饰图形，例如标注坐标轴、设置视角、设置光源、改变着色模式等(此步骤可以默认使用 MATLAB 提供的默认值)。

在绘图时，准备数据的工作一般是根据实际需要完成，创建图形输出窗口一般可用绘图函数自动创建，所以首先介绍创建三维图形的一般命令。

1. 三维线性图形

plot3()命令将绘制二维图形的函数 plot()特性扩展到了三维空间。函数格式除了包括第三维的信息(如 z 方向)之外，其他与二维函数 plot()相同。一般语法调用格式是：

```
plot3(x1,y1,z1,s1,x2,y2,z2,s2,…)
```

其中 xn、yn 和 zn 是向量或矩阵，sn 是可选的字符串，用来指定颜色、标记符号和线形。如果省略，MATLAB 将会按照默认值给出设置。

总的来说，plot3()可用来画一个单变量的三维函数，如下例所示，该例将绘制一串宝石项链。

```
t=(0:0.02:2)*pi;
x=sin(t);
y=cos(t);
z=cos(2*t);
plot3(x,y,z,'b-',x,y,z,'bd');  绘制三维线性图。
view([-82,58]);                %设置视角。
box on;
legend('链','宝石');        %图例标注。
```

输出的结果如图 4.24 所示。

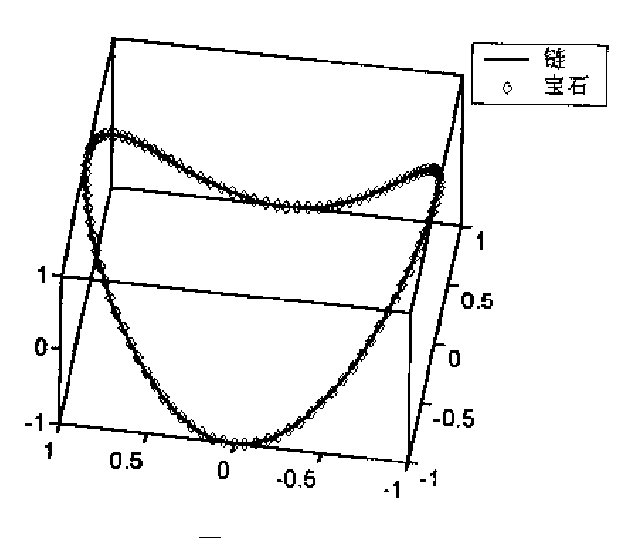


图 4.24 宝石项链

通过指定 plot3()命令的多个参量(如上例)或使用 hold 命令，可以把多条直线或曲线重叠画出。例如，增加维数的 plot3()命令可以使多个二维图形沿一个轴排列起来，而不是直接将二维图形叠到另一个上面。

```
% 二维图形在三维空间的排列。
x=linspace(0,3*pi);  %x 轴的数据。
z1=sin(x);  z2=sin(2*x);  z3=sin(3*x);       %在 x-z 平面内画图。
y1=zeros(size(x));  y3=zeros(size(x));  y2=y3/2;  %变量的预定义。
plot3(x,y1,z1,x,y2,z2,x,y3,z3);
grid;  xlabel('x-axis');  ylabel('y-axis');  zlabel('z-axis');
title('sin(x),sin(2x),sin(3x)');
```

输出的结果如图 4.25 所示。

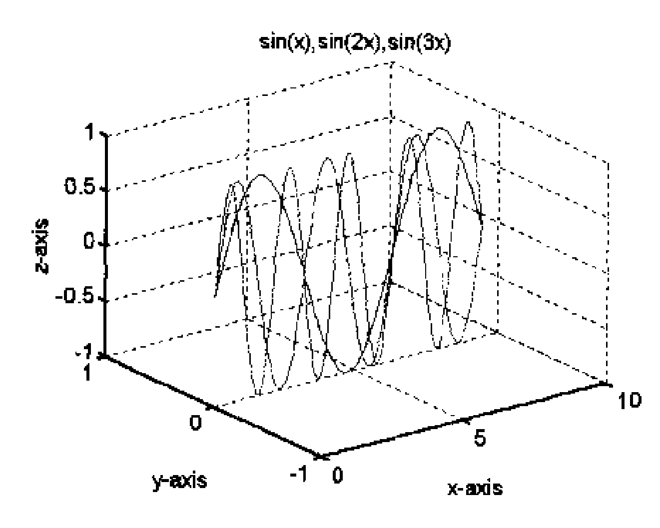


图 4.25 正弦曲线图

2. 三维曲面

- 平面网格点的生成

在数学上，函数 z=f(x,y)的图形是三维空间的曲面，在 MATLAB 中，总是假设函数 z=f(x,y)是定义在一个矩形的区域 D=[xmin,xmax]×[ymin,ymax]上的。为了在区域 D 上绘制三维曲面，MATLAB 首先将[xmin,xmax]在 x 方向分成 m 份，将[ymin,ymax]在 y 方向分成 n 份，由各划分点分别作平行于坐标轴的直线，将区域 D 分成 m×n 个小矩形块，计算出在网格点的函数值。对于每个小矩形，在空间中决定出 4 个顶点(xi,yi,f(xi,yi))，连接 4 个顶点得到一个空间中的四边形片。所有这些四边形片一起构成函数 z=f(x,y)定义在区域 D 上的空间网格曲面。

为方便起见，MATLAB 用函数 meshgrid 来生成 x-y 平面上的小矩形顶点坐标值的矩阵。函数 meshgrid 的调用形式如下：

```
[X,Y]=meshgrid(x,y)
```

或是

```
[X,Y]=meshgrid(x)
```

第 2 种形式等价于 meshgrid(x,x) 。这里，x 是区间[xmin,xmax]上划分点组成的向量，而 y 是区间[ymin,ymax]上划分点组成的向量。输出变量 X 和 Y 都是矩阵，矩阵 X 的行向量都是向量 x，Y 的列向量都是向量 y。于是 X 和 Y 的元素组(X(i,j),Y(i,j))恰好是区域 D 的第(i,j) 网格顶点。换句话说，函数 meshgrid()将由两个向量决定

的区域转换成网格点矩阵。例如：

```
%定义两个行向量。
x=[1,2,3]
x =
     1     2     3
y=[0,1,2]
y =
     0     1     2
%由x和y生成平面网格点矩阵。
[X,Y]=meshgrid(x,y)
X =
     1     2     3
     1     2     3
     1     2     3
Y =
     0     0     0
     1     1     1
     2     2     2
```

- 曲面网线图和网面图绘图

如果要画一个三维的曲面，MATLAB将以meshgrid配合mesh或surf命令来绘图。绘图一般分为以下几个步骤：

(1) 用函数meshgrid()生成平面网格点矩阵[X,Y]= meshgrid(x,y)。

(2) 由[X,Y]计算函数值矩阵Z，即是对xy平面上的点计算出相应的值z。

(3) 用函数mesh()绘制曲面网线图，用函数surf()绘制曲面网面图。

函数mesh()的常用语法格式如下：

- mesh(X,Y,Z)　由3个矩阵X、Y、Z绘制空间网线图，即以矩阵元素(Xij,Yij,Zij)为空间坐标在空间描点作图，点与点之间用线段连接，线的颜色随Z值即空间高度的不同而变化，也可通过colormap()函数指定网格线的颜色。
- mesh(x,y,Z)　由n维向量x，m维向量y对m×n矩阵Z绘制网线图，坐标为(x(i),y(j),Z(i,j))。
- mesh(Z)　以矩阵Z中元素的下标为x、y值，相应元素值为z制作矩阵Z的空间网格图，即x=1:n、y=1:m，也就是把xy平面等距的划分为m×n个小矩形单元并作图。
- mesh(…,C)　C为颜色矩阵。

函数surf()的常用语法格式类似于函数mesh()，区别在于surf()函数对网线间的网线元进行填充，形成表面图。

下面是这两个函数的例子。4个图形中分别绘制了椭圆抛物面、马鞍面、阔边帽面、峰形函数的网线图和网面图。

- 椭圆抛物面

```
x=-4:0.2:4;
y=x;
[X,Y]=meshgrid(x,y);     %生成 x-y 坐标“格点”矩阵。
Z=X.^2/9+Y.^2/9;         %计算格点上的函数值。
```

```
subplot(1,2,1);
mesh(X,Y,Z)
title('椭圆抛物面网线图')
subplot(1,2,2);
surf(X,Y,Z)
title('椭圆抛物面网面图')
```

其结果如图 4.26 所示。

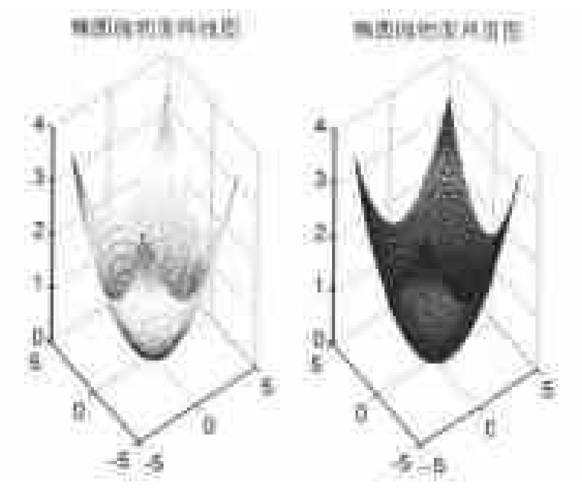


图 4.26 椭圆抛物面

- 马鞍面

```
x=-4:0.2:4;
y=x;
[X,Y]=meshgrid(x,y);        %生成 x-y 坐标“格点”矩阵。
Z=X.^2/9-Y.^2/9;            %计算格点上的函数值。
subplot(1,2,1);
mesh(X,Y,Z)
title('马鞍面网线图')
subplot(1,2,2);
surf(X,Y,Z)
title('马鞍面网面图')
```

其结果如图 4.27 所示。

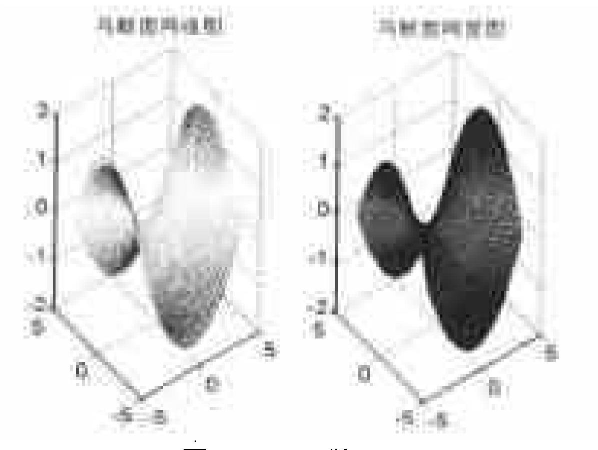
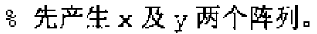


图 4.27 马鞍面

● 阔边帽面

```
x=-7.5:0.5:7.5;
y=x;                        % 先产生 x 及 y 两个阵列。
[X,Y]=meshgrid(x,y);        % 再以 meshgrid 形成二维的网格数据。
R=sqrt(X.^2+Y.^2)+eps;      % 加上 eps 可避免当 R 在分母趋近零时会无法定义。
Z=sin(R)./R;                % 产生 z 轴的数据。
subplot(1,2,1);
mesh(X,Y,Z)                 % 将 z 轴的变化值用网格方式画出。
title('阔边帽面网线图')
subplot(1,2,2);
surf(X,Y,Z)
title('阔边帽面网面图')
```



其结果如图 4.28 所示。

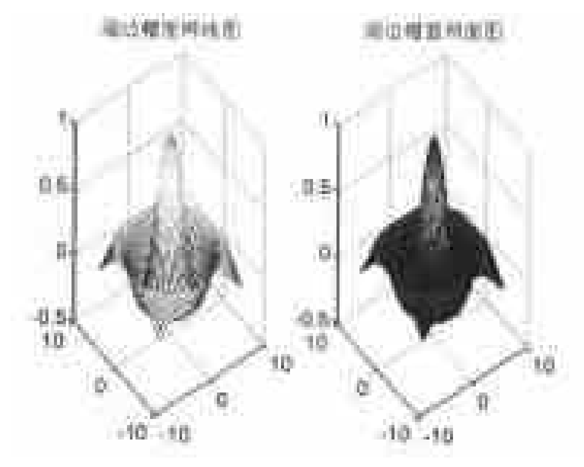


图 4.28 阔边帽面

● 峰形图

peaks 函数是 MATLAB 为方便测试立体绘图提供的，该函数在 MATLAB 中表示为：

```
z=3*(1-x).^2.*exp(-(x.^2)-(y+1).^2)-10*(x/5-x.^3-y.^5).*exp(-x.^2-y.
^2)-1/3*exp(-(x+1).^2-y.^2)
```

该函数可产生一个有致的曲面，包含 3 个局部极大点及 3 个局部极小点。

```
subplot(1,2,1);
mesh(peaks) % 直接将已定义的 peaks 函数用网格方式画出。
title('峰形函数的网线图')
subplot(1,2,2);
surf(peaks)
title('峰形函数的网面图')
```

其结果如图 4.29 所示。

● 等高线图绘制函数

与三维绘图有关的还有等高线图(又称等值线)，等高线图在地理、工程等方面应用很广泛，其相关命令为 contour,contour3。contour 是将等值线图用二维图表示，其语法格式有以下几种：

- contour(Z) 其中 Z 是一个二维矩阵，该命令绘制矩阵 Z 表示的二元函数的等高线，即绘制曲面上点坐标为(i,j,Zij)的等高线。
- contour(Z,n) 表示指定的等高线条数，默认值为 10。
- contour(Z,v) 按向量 v 的元素指定的值绘制相应得的等高线，每条等高线对应一个 v(i)，等高线的条数为向量 v 的维数。
- contour(X,Y,Z)类似于 mesh()函数中的说明，当 x,y 为向量时，由 x,y 确定 xy 平面坐标轴的范围；当 X,Y 为矩阵时，绘制以矩阵元素(Xij,Yij,Zij)为空间坐标的曲面的等高线图。进一步来说，contour(X,Y,Z,...)可在变量的后面加 n、v、linespec 等参数。

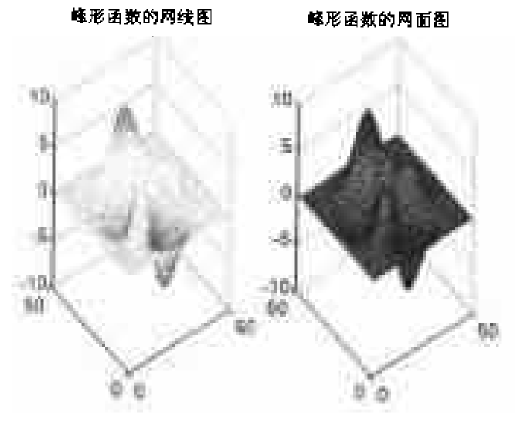


图 4.29 峰形函数图

contour3 则是将等值线用三维图表示，其语法与 contour 类似，只是将对应的关键字 contour 改成 contour3，其余部分相同，这里不再重复。

该类函数的操作如下例所示，图 4.30 给出了函数二维和三维的等高线图。

```
[X,Y,Z]=peaks;          % x,y 及 z 轴的数据由 peaks 函数定义。
subplot(2,2,1)
contour(Z,20)           % 画出 peaks 的 z 轴二维等值线图，20 为等值线的数目。
subplot(2,2,2)
contour(X,Y,Z,20)       % 画出 peaks 的二维等值线图，注意 x,y 轴与上图不同。
subplot(2,2,3)
contour3(Z,20)          % 画出 peaks 的 z 轴二维等值线图。
subplot(2,2,4)
contour3(X,Y,Z,20)      % 画出 peaks 的三维等值线图，注意 x,y 轴与上图不同。
```

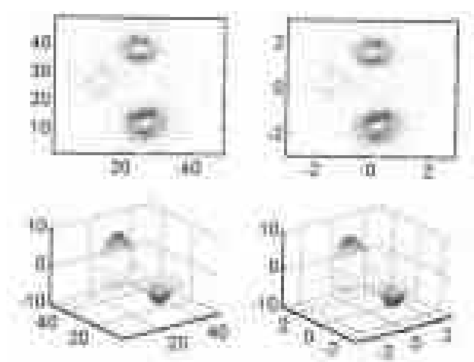


图 4.30 等高线图

除直角坐标系外，我们还可以在极坐标系中绘制等高线，通过 pol2cart()函数把数据从极坐标转化为直角坐标。pol2cart()函数的使用在此不作介绍。

- 其他几种函数用法

  在 MATLAB 中，三维函数还有 meshc()、meshz()、surfc()、surfl()、contourf()、waterfall()等，其变量的说明类似相应的 mesh()函数、surf()函数、contour()函数，这里只给出例子，详细的变量说明请参见相应的函数说明及帮助。

  ◆ meshc() 该函数同时绘出曲面的网格线图和等高线。

  该函数绘图的操作参见下例，结果如图 4.31 所示。

```
[x,y,z]=peaks;
meshc(x,y,z);
axis([-inf inf -inf inf -inf inf]);
title('用 meshc 函数绘图');
```

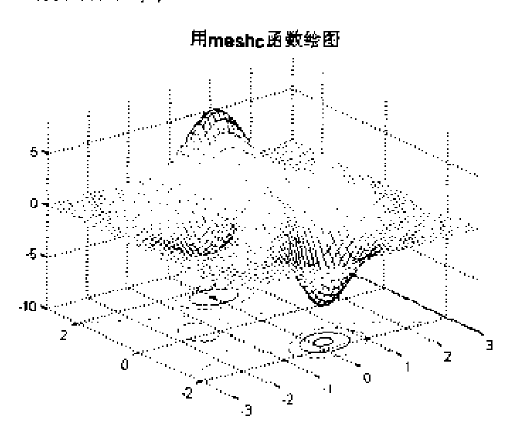


图 4.31 用meshc函数绘图

◆ meshz() 该函数绘出曲面和零平面。

利用函数绘图的操作参见下例，结果如图 4.32 所示。

```
[x,y,z]=peaks;
meshz(x,y,z);
axis([-inf inf -inf inf -inf inf]);
title('用 meshz 函数绘图');
```

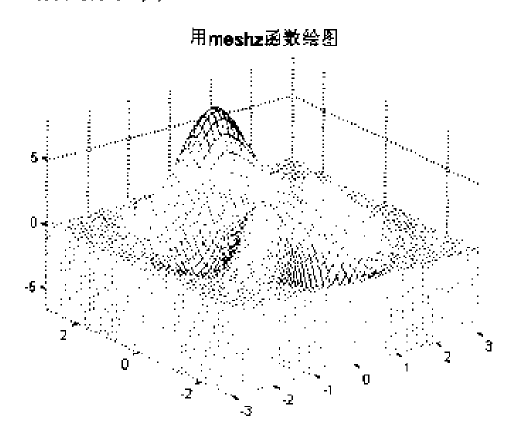


图 4.32 用meshz函数绘图

◆ surfc() 利用该函数同时绘出表面图和等高线，结果如图 4.33 所示。

```
[x,y,z]=peaks;
surfc(x,y,z);
axis([-inf inf -inf inf -inf inf]);
title('用 surfc 函数绘图');
```

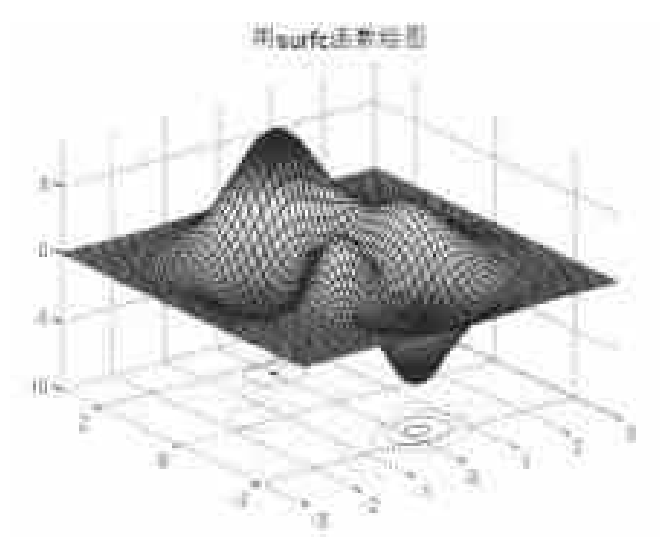


图 4.33 用surfc函数绘图

◆ surfl() 该函数给出带光照效果的彩色表面图。

下例是有关的应用，其结果如图 4.34 所示。

```
x=-25:1.25:25;
y=-25:1.25:25;
[xx,yy]=meshgrid(x,y);
z=xx.^2./9-yy.^2./4+eps;
surfl(xx,yy,z);
colormap(gray);
title('用 surfl 函数绘图');
```

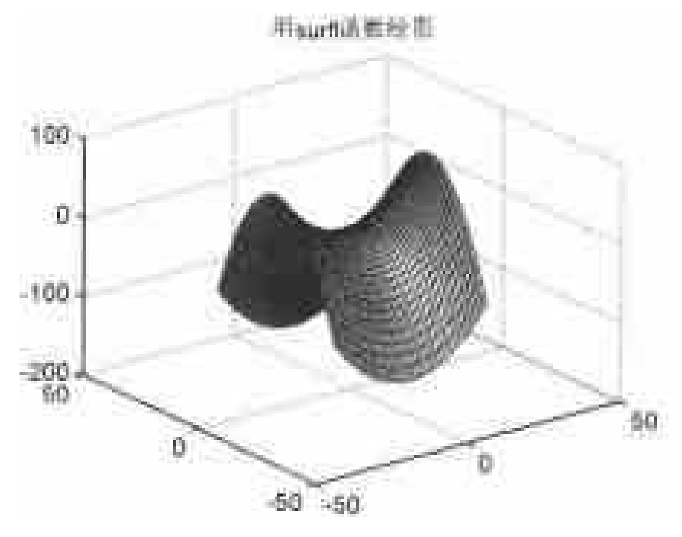


图 4.34 用surfl函数绘图

◆ waterfall() 该函数使图形产生水流效果，结果如图 4.35 所示。

waterfall 可在 x 方向或 y 方向产生水流效果。

下列命令将在 x 方向产生水流效果：

```
[x,y,z]=peaks;
waterfall(x,y,z);
axis([-inf inf -inf inf -inf inf]);
title('x 方向水流效果的阔边帽图')
```

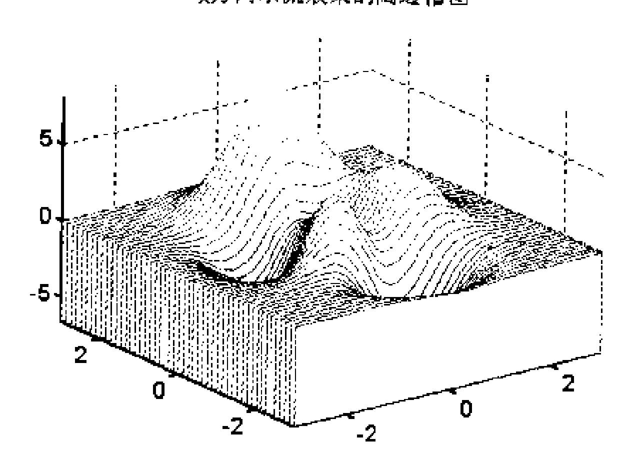


图 4.35　x方向水流效果的阔边帽图

下列命令产生 y 方向的水流效果：

```
[x,y,z]=peaks;
waterfall(x',y',z');
axis([-inf inf -inf inf -inf inf]);
title('y 方向水流效果的阔边帽图')
```

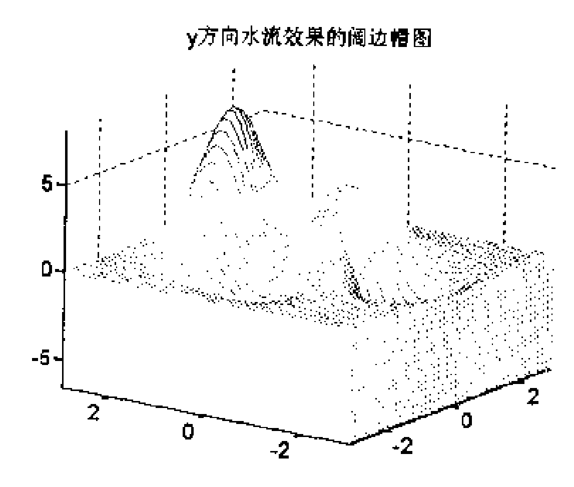


图 4.36　y方向水流效果的阔边帽图

3. 三维图形的修饰

三维图形的修饰类似二维图形，同样有 title()、legend()、text()、color()、axis()等函数，对轴的标注除 xlabel()、ylabel()函数外，增加了 zlabel()函数以标注 z 轴。对这些函数的使用，在此不作介绍，请参看 4.1.2 节的内容。

4.2.2 三维图形的特殊处理

三维图形的特殊处理包括设置视角、设置光照效果以及图形的消隐、透视、镂空和裁切等，下面分别进行介绍。

- 设置视角
  - 先了解视角原理。当观察空间三维图形时，从不同的方向观察，看到的效果不同。这关系到视角的设置，确定视角就是确定在三维空间中观察物体的方位。确定视角一般可分为两种情况，一种是指定观察的方位角和仰视角，另一种是指定观察点。如图 4.37 所示，方位角指的是以 y 轴的负半轴起在 xy 平面视线在平面上的投影角度，即图中的角 a。俯视角是视线在 xy 平面的投影与视线的夹角，即图中的角 s。确定观察点即是指出视线方向直线上任意点的坐标，其具体数值的意义不大，例如(1,1,1)与(2,2,2)同是指定同一个视线方向。显而易见，指定观察点也就指定了方位角与俯视角。

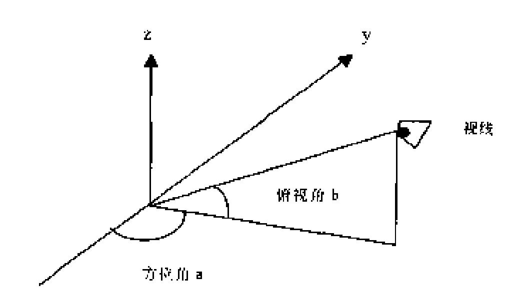


图 4.37 视角定义

  - 设置视角函数 view()。MATLAB 提供了 view()函数以设置三维图形的视角，该函数的常用语法格式为：

    view(az,el) 设置三维图形的视角，其中 az(azimush)为观察的方位角，el(elevation)为仰视角。

    view([x,y,z]) 由坐标(x,y,z)指定观察点，即确定视角。此时视角只与矢量(x,y,z)的方向有关，而与矢量的长度无关。

    view(3) 设置三维图形视角默认值 az=-37.5、el=30，此前绘制的三维图形均是此视角。

    view(T) 由 4×4 的转化矩阵确定视角。

下例是一个峰形图在不同视角中看到的结果，如图 4.38 所示。

```
subplot(2,2,1);
mesh(peaks)
view(-37.5,30)
title('az=-37.5、el=30')
subplot(2,2,2);
mesh(peaks)
view(-10,15)
title('az=-10、el=15')
subplot(2,2,3);
```

```
mesh(peaks)
view(30,-37.5)
title('az=30、el=-37.5')
subplot(2,2,4);
mesh(peaks)
view(15,-10)
title('az=15、el=-10')
```

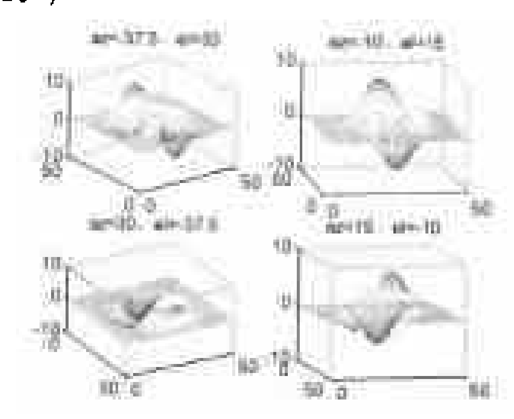


图 4.38 设置不同的视角

- 设置光照效果

  添加光照效果，可以更加真实地显示三维图形。在 MATLAB 中，要设置光照效果，首先应通过 light()函数创建光照对象，然后通过模拟物体在自然光条件下的光影效果调整图形颜色。创建光照对象相当于创建一个光源，该光源的特性由光照对象的属性确定，包括光源发出光的颜色(color)、光源的形状(style)及光源的位置(position)。

  基于漫射、镜面反光和环境照明等原理，使用插值彩色可以绘出效果很好的曲面，并且环境照明、漫射反射、镜面反光对视光效果的相对贡献可以用变量 K 表示，其中 K=[ka,kd,ks,spread]，默认值为[0.55 0.6 0.4 10]。下例将给出附加光照效果的 peaks 曲面模型，结果如图 4.39 所示。

```
[X,Y,Z]=peaks(30);
surfl(X,Y,Z)
colormap(cool);      %使用冷色调的色图矩阵
title('使用默认的光照效果')
```

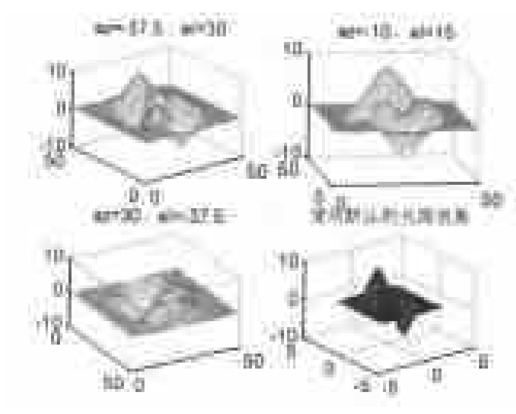


图 4.39 设置光照效果

```
shading interp; %使用插值彩色
surfl(X,Y,Z,[-90 30],[0.5 0.6 1.5 10]);% 改变K因子，不使用插值着色
title('改变光照效果')
```

改变光照效果的示意图如图 4.40 所示。

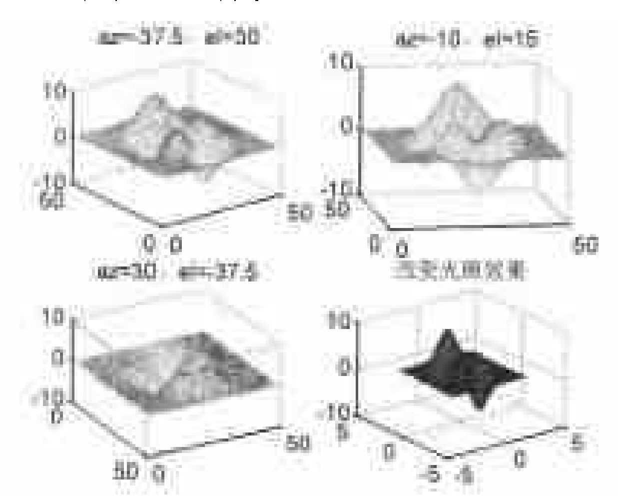


图 4.40 改变光照效果

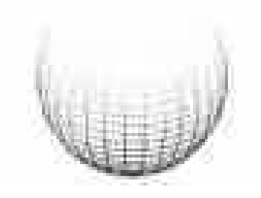
- 图形的消隐与透视

  在三维空间中绘制多个图形时，由于图形之间要相互遮盖，就涉及到消隐与透视问题。消隐是指图形相互重叠的部分不再显示，透视是指相互重叠的部分互不妨碍，全面显示。MATLAB 提供了 hidden 函数实现该功能。

  - **hidden on** 图形间消隐，为默认值。
  - **hidden off** 图形间透视。

该函数的应用如下例所示，结果如图 4.41 所示。





透视图　　消隐图

图 4.41 透视图与消隐图

```
[X0,Y0,Z0]=sphere(30);                %产生单位球面的三维坐标。
X=2*X0;Y=2*Y0;Z=2*Z0;                 %产生半径为 2 的球面的三维坐标。
clf,subplot(1,2,1);
surf(X0,Y0,Z0);                       %画单位球面。
shading interp                        %采用插补明暗处理。
hold on,mesh(X,Y,Z),colormap(hot),hold off        %采用 hot 色图。
hidden off                            %产生透视效果。
axis equal,axis off                   %不显示坐标轴。
title('透视图')
subplot(1,2,2);
surf(X0,Y0,Z0);                       %画单位球面。
```

```
shading interp                    %采用插补明暗处理。
hold on,mesh(X,Y,Z),colormap(hot),hold off        %采用 hot 色图。
hidden on                         %产生消隐效果。
axis equal,axis off               %不显示坐标轴。
title('消隐图')
```

- 镂空和裁切

  将图形中的某些区域赋值为非数可以达到镂空图形的目的，同理也可以通过将图形中的某些区域赋值为零，达到裁切图形的目的。下面通过具体例子来说明。

  - 利用“非数”NaN，对图形进行镂空处理。

```
clf,P=peaks(30);P(18:20,9:15)=NaN;        %镂空
surfc(P);colormap(summer)
light('position',[50,-10,5]),lighting flat
material([0.9,0.9,0.6,15,0.4])
```

  结果如图 4.42 所示。

  - 利用赋值零，对图形进行裁切处理。

```
clf,x=[-8:0.2:8];y=x;[X,Y]=meshgrid(x,y);ZZ=X.^2-Y.^2;
ii=find(abs(X)>6|abs(Y)>6);%确定超出[-6,6]范围的格点下标。
ZZ(ii)=zeros(size(ii));           %强制为 0。
surf(X,Y,ZZ),shading interp;colormap(copper)
light('position',[0,-15,1]);lighting phong
material([0.8,0.8,0.5,10,0.5])
```

经裁切处理的图形如图 4.43 所示。

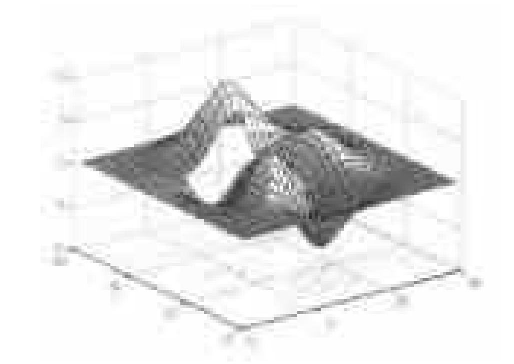


图 4.42　镂方孔的曲面

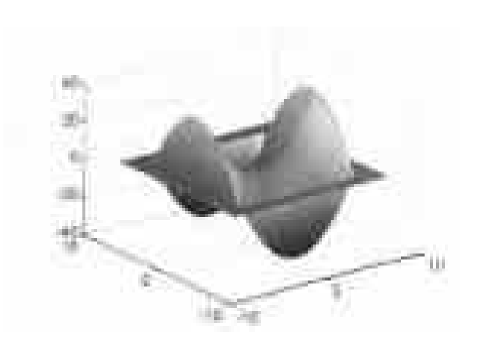


图 4.43　经裁切处理的图形

4.2.3　一些特殊的三维图形

在介绍二维图形时，讨论了一些特殊的二维图形，如直方图、饼图、柄图、向量场图(即二维箭图)、极坐标图等，相对应的三维图形中也有一些特殊图形，其用法与二维特殊图形类似，因此在下面的介绍中只给出简要的说明及例子，详细情况请参见二维特殊图形及三维图形的一般绘制方法。

- 三维直方图

  绘制三维直方图的函数有 bar3()和 bar3h()，下面是这两个函数的应用，结果如图 4.44 所示。

```
clf;x=-2:2;                                      %注意：自变量要单调变化。
Y=[3,5,2,4,1;3,4,5,2,1;5,4,3,2,5];               %各因素的相对贡献份额。
subplot(1,2,1),bar3(x',Y',1)                     %“队列式”直方图。
xlabel('因素ABC'),ylabel('x'),zlabel('y')
colormap(summer)                                 %控制直方图的用色。
subplot(1,2,2),bar3h(x',Y','grouped')            %“分组式”水平直方图。
ylabel('y'),zlabel('x')
```

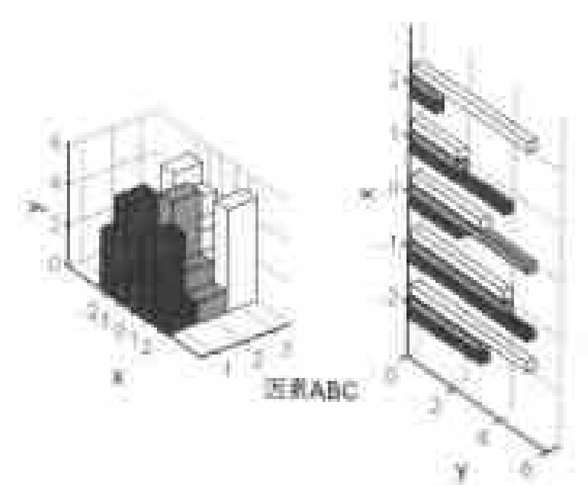


图 4.44 三维直方图

- 三维柄图

三维柄图也称三维火柴杆图或三维针状图，用函数 stem3()绘制，如图 4.45 所示。

```
% 一个离散方波的快速 Fourier 变换的幅频。
th = (0:127)/128*2*pi;                           %角度采样点。
rho=ones(size(th));                              %单位半径。
x = cos(th);y = sin(th);
f = abs(fft(ones(10,1),128));                    %对离散方波进行 FFT 变换，并取幅值。
rho=ones(size(th))+f';                           %取单位圆为绘制幅频谱的基准。
subplot(1,2,1),polar(th,rho,'r')
subplot(1,2,2),stem3(x,y,f','d','fill')          %取菱形离散杆头，并填色。
view([-65 30])                                   %控制角度，为表现效果。
```

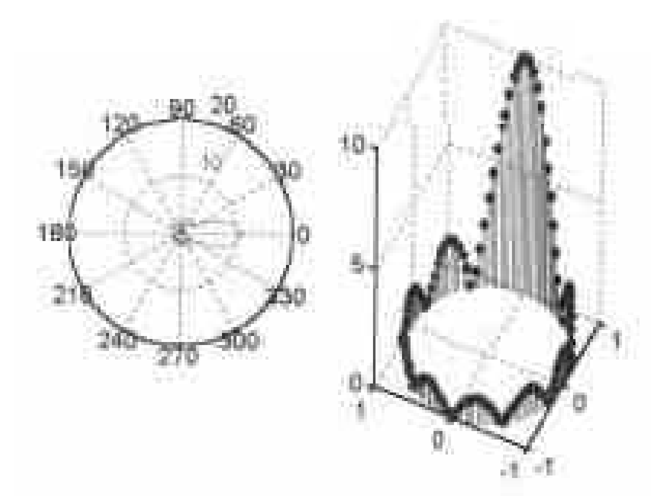


图 4.45 离散方波FFT的幅频

- 三维饼图

绘制三维饼图的函数为 pie3()，其应用可参见下例最后的结果如图 4.46 所示。

```
a=[1,1.6,1.2,0.8,2.1];
subplot(1,2,1),pie(a,[1 0 1 0 0]);
legend({'1','2','3','4','5'});
subplot(1,2,2),pie3(a,a==min(a));
colormap(cool);
```

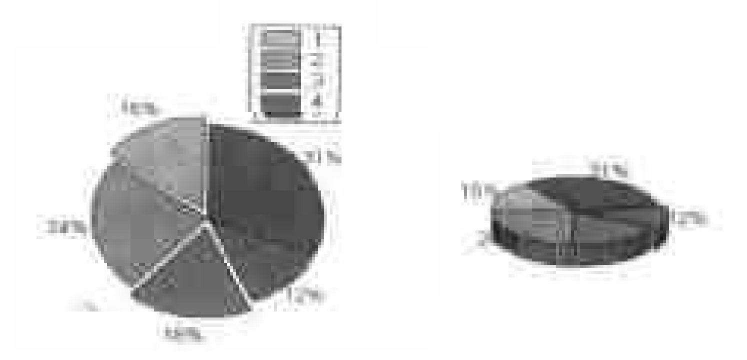


图 4.46 饼形统计图

- 三维箭图

quiver3 用于绘制三维矢量图，如空中的飞行轨迹、三维空间的静电场或重力场等，其应用如下例，结果如图 4.47 所示。

```
clf;
t=0:10;
quiver3(2*t,2*t,3*t,2,2,3)
```

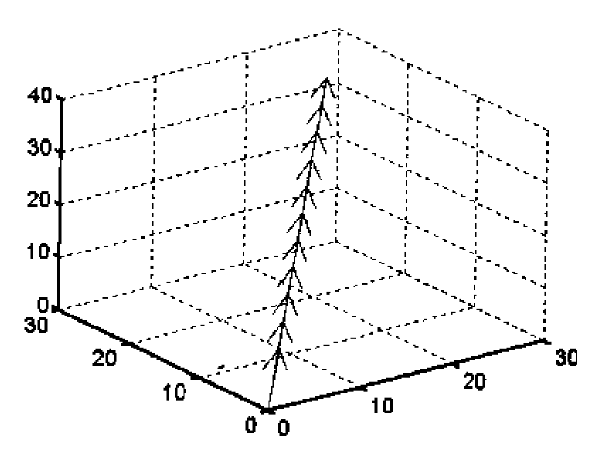


图 4.47 三维箭图

第5章　MATLAB 6常用图像操作

在社会生活和科学研究中，人们经常要与图像打交道。图像信息是人类认识世界的主要知识来源，国外学者的统计结果表明，人类所获得的外界信息有70%以上是通过眼睛获得的。“百闻不如一见”说明了在很多场合里，其他形式的信息都比不上图像所传达的信息丰富和真切。

数字图像处理是使用数字计算机对图像进行加工和处理的过程，最早的使用是二十世纪60年代美国喷气推进实验室(Jet Propulsion Laboratory)使用数字计算机对大批月球照片处理，得到了清晰的图像。此后，数字图像处理技术在各个领域都得到了广泛的应用。主要包括：遥感中的资源探测、气象预报以及军事侦察，在生物医学领域运用于细胞分类和计算机层析摄影(CT)，在工业生产中用于加工、装配和拆卸及质量检验，在军事和公安方面用于武器制导和指纹识别等。

数字图像研究的领域非常广泛，从学科上可以分为图像的数字化、图像变换、图像增强、图像恢复、图像分割、图像分析和理解，以及图像的压缩等。

MATLAB图像处理工具箱提供了丰富的图像处理函数，主要可以完成以下功能：

- 图像的几何操作
- 图像的邻域和图像块操作
- 线性滤波和滤波器设计
- 图像变换
- 图像分析和增强
- 二值图像操作
- 感兴趣区域处理

MATLAB图像工具箱提供的函数大多数是M文件，我们可以查看这些文件的代码并进行改进，也可以把自己编写的代码加入其中，来扩充图像处理的功能。

MATLAB 6提供的图像处理工具箱增加了如下特性：支持16位图像操作；对一些函数进行了优化，提高了执行速度，比如：bwfill、bwselect、bwlabel、dilate、erode、histeq、imresize、imrotate、ordfilt2、medfilt2、和im2uint8；为medfilt 2和ordfilt2提供了边界补零选项；并且提供了一个新的函数 im2uint16，可以把图像转化为uint16类型。

另外，随着图像处理技术的发展，小波分析在图像处理中的应用也越来越广泛，MATLAB小波分析工具箱提供了很多有用的函数，我们将在第6章中介绍。

5.1　MATLAB 中图像类型转换

MATLAB 中的数字图像是由一个或多个矩阵表示的，矩阵可以是实数，也可以是复数。这意味着 MATLAB 强大的矩阵运算用于图像处理非常合适，矩阵运算的语法对 MATLAB 中的数字图像同样适用。

在 MATLAB 中，图像是按像素存储的，即矩阵的每个元素代表一个像素。例如一幅 200 行 300 列的图像，在 MATLAB 中存储为 200×300 大小的矩阵。有些图像，如 RGB 图像，需要三维矩阵表示，每一维代表一种颜色，这样一幅 200 行 300 列的 RGB 图像就需要用 200×300×3 的矩阵表示。

如果不加说明，MATLAB 中图像数据矩阵的存储方式为双精度(double)类型，即 64 位的浮点数。由于几乎所有的 MATLAB 函数及其工具箱函数都可以使用 double 作为参数类型，这种存储方法在使用中不需要进行数据类型转换。但是对于存储图像来说，用 64bit 来表示图像数据存储量特别巨大，所以 MATLAB 还支持图像数据的无符号整型(uint8)存储方式，即图像矩阵中的每个数据占用一个字节。由于 MATLAB 及工具箱中的大多数运算和函数(比如最基本的矩阵加减运算)都不支持 uint8 类型，在运算时通常要将图像转换成 double 型。uint8 类型的优势仅在于节省存储空间。

由于 MATLAB 图像处理中存在 uint8 与 double 两种图像数据类型，所以在使用工具箱函数时要按照函数要求输入的参数类型传递参数。而且由于 uint8 与 double 两种类型数据的值域不同，使用时还要注意输出数据值域的转换。

5.1.1　MATLAB 图像处理工具箱支持的图像类型

图像类型是指图像在 MATLAB 数据文件中的存储方式。MATLAB 图像处理工具箱支持 4 种图像类型，它们是：

- 真彩色图像
- 索引色图像
- 灰度图像
- 二值图像

另外，MATLAB 还可以处理由多帧图像组成的图像序列。

1. 真彩色图像

真彩色图像又称 RGB 图像，它是利用 R、G、B 3 个分量表示一个像素的颜色，R、G、B 分别代表红、绿、蓝 3 种不同的颜色，通过三基色可以合成出任意颜色。所以对一个尺寸为 m×n 的彩色图像来说，在 MATLAB 中则存储为一个 m×n×3 的多维数组。如果需要知道图像 A 中(x，y)处的像素 RGB 值，则可以使用这样的代码 A(x,y,1:3)。

真彩色图像可用双精度型来存储，此时亮度值的范围是[0,1]，一个像素值为(0,0,0)代表黑色，值为(1,1,1)代表白色，这一点与 Windows 编程规则不同。有的图像文件格式用 24 位

存储 RGB 图像，红、绿、蓝各占 8 位，这样就可以有 2^{24} =16777216 种颜色。真彩色图像也可以用无符号整型来存储，如果用无符号整型存储 RGB 图像，则亮度值的范围为[0，255]。

图 5.1 是一幅真彩色图像的结构。

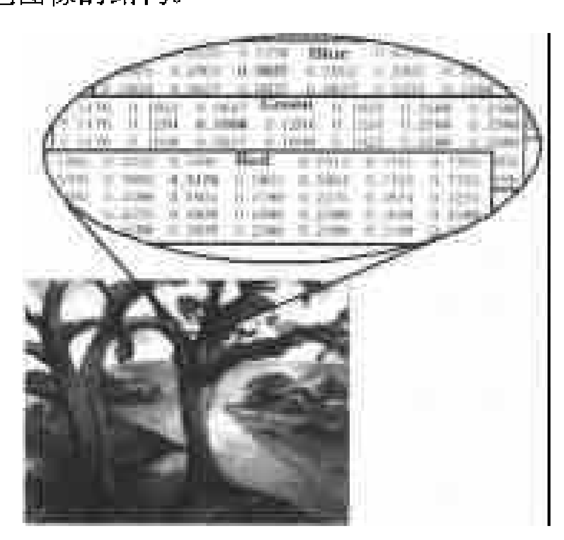


图 5.1 真彩色图像的存储结构

2. 索引色图像

索引色图像把不同的颜色对应为不同的序号，各像素存储的是颜色的序号而不是颜色本身。MATLAB 中的索引色图像包含两个结构，一个是调色板，一个是图像数据矩阵。调色板是一个 m×3 的色彩映射矩阵，矩阵的每一行都代表一种色彩，与真彩色图像相同，通过 3 个分别代表红、绿、蓝颜色强度的双精度数，形成一种特定的颜色。调色板矩阵每个元素的值可以是在[0,1]之间的双精度浮点数。

索引色图像数据矩阵的类型可以是 double 类型或者 uint8 类型。图像矩阵和调色板序号之间的关系取决于图像矩阵的类型：当图像数据为 double 类型时，值 1 代表调色板中的第 1 行，值 2 代表第 2 行，以此类推；当图像数据是 uint8 类型，0 则代表调色板的第 1 行，1 代表第 2 行，以此类推，要注意这一区别，否则得到的图像会有所差别。调色板通常和索引图像存在一起，当读入图像时，MATLAB 同时加载调色板和图像。MATLAB 提供了一些函数可以产生预存的标准调色板的函数。

默认情况下，调用调色板函数会产生一个 64×3 的调色板，当然，用户也可以制定调色板的大小。如 hot(m)产生一个 m×3 的调色板，其颜色范围从黑经过红、橘红、黄到白。

3. 灰度图像

灰度图像就是只有强度信息，而没有颜色信息的图像，它在早期的图像处理中占有很重要的地位。存储灰度图像只需要一个数据矩阵，矩阵的每个元素表示对应位置的像素的灰度值。灰度图像的数据类型可以是 double 类型，这时值域为[0，1]；也可以是 uint8 类型，值域为[0，255]。

尽管灰度图像从来不和调色板一起存储，但是 MATLAB 还是要利用调色板来显示灰度图像。

图 5.2 是一幅索引色图像的结构。

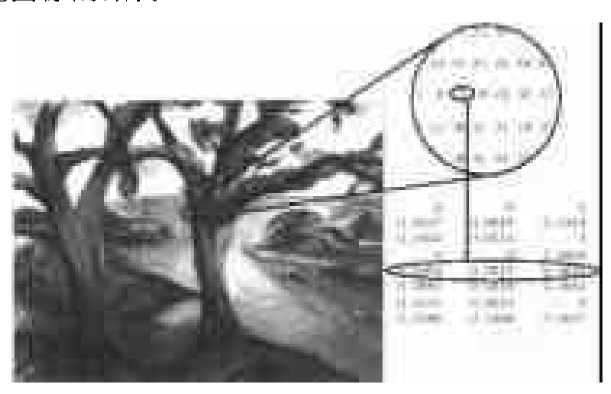


图 5.2 索引色图像的存储结构

图 5.3 是一幅灰度图像的结构。

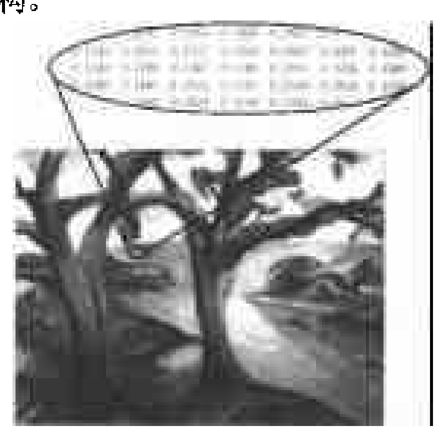


图 5.3 灰度图像的存储结构

4. 二值图像

二值图像就是只有黑白两种值的图像，我们可以把它看作是特殊的灰度图像。二值图像只需一个数据矩阵来存储，每个像素只取 0 或者 1。二值图像可以采用 uint8 或 double 类型存储，但是由于 uint8 类型节省空间，通常都使用 uint8 类型。

MATLAB 图像处理工具箱返回二值图像的函数都把返回结果作为 uint8 类型的逻辑矩阵，并用一个逻辑标志来表示值域范围。如果逻辑标志为真，则返回结果为[0,1]。如果逻辑标志为假，返回结果为[0,255]。图 5.4 是一幅二值图像的结构。

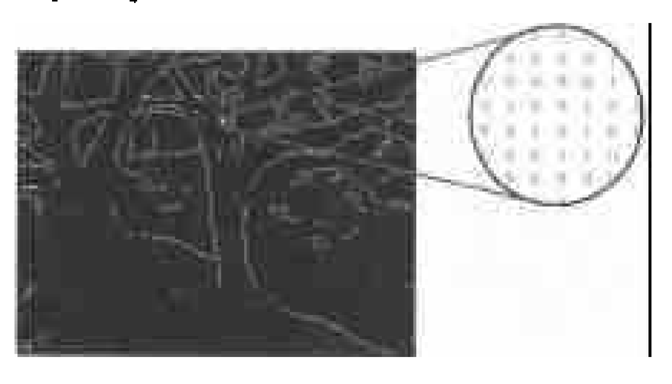


图 5.4 二值图像的存储结构

5. 图像序列

有些情况下，需要一系列按照时间或者序号排列起来的一组图像，例如核磁共振图像片(MRI)或者电影文件。

图像处理工具箱中定义了函数可以将多帧图像连接成图像序列。图像序列是一个四维的数组，图像的长、宽、颜色深度构成图像的前三维，图像帧的序号构成第四维。比如一个包含了 4 幅 500×400 真彩色图像的序列，图像的大小则为 500×400×3×4。

MATLAB 的 cat 函数可以将分散的图像合并成图像序列，但是要求各图像的尺寸必须相同，而且如果是索引色图像，各图像的调色板也必须是一样的。比如要将 A1、A2、A3、A4、A5 5 幅图像合并成一个图像序列 A，可以用下面的语句实现：

```
A=cat(4,A1,A2,A3,A4,A5)
```

另外，也可以从图像序列中单独抽出一帧或者几帧，比如：

```
FRM 3=MULTI(:,:,:,3)
```

该命令表示将序列 MULTI 中的第 3 帧抽出来并赋给矩阵 FRM 3。

5.1.2 转换图像类型

在有些图像操作中，需要对图像的类型进行转换。比如要对一幅索引色图像滤波，首先应该将它转换成真彩色图像或者灰度图像，这时 MATLAB 将会对图像的灰度进行滤波，这正是通常意义上的滤波。如果不将索引色图像进行转换，MATLAB 则对图像调色板的序号进行滤波，这是没有意义的。下面对一些 MATLAB 图像处理工具箱中常用的类型转换函数进行介绍。

1. dither

dither 函数通过抖动算法转换图像类型，其语法格式为：

```
X=dither(RGB,map)
X=dither(RGB,map,Qm,Qe)
BW=dither(I)
```

这里 X=dither(I1,map)通过抖动算法将真彩色图像 RGB 按指定的调色板 map 转换成索引色图像 X。

X=dither(RGB,map,Qm,Qe)利用给定的参数 Qm,Qe 从真彩色图像 RGB 中产生索引色图像 X。Qm 对于补色决定各颜色轴的量化位数，Qe 决定量化误差的位数。如果 Qe<Qm，则不进行抖动操作。Qm 的默认值是 5，Qe 的默认值是 8。

BW=dither(I2)将灰度图像 I 抖动成二值图像 BW。

输入图像可以是 double 或 uint 8 类型，如果输出的图像是二值图像或颜色种类不超过 256 的索引色图像，则是 uint 8 类型，否则为 double 型。

2.　im2bw

im2bw 函数通过设置亮度阈值将真彩色、索引色、灰度图转换成二值图，其语法格式为：

```
BW=im2bw(I,level)
BW=im2bw(X,map,level)
BW=im2bw(RGB,level)
```

BW=im2bw(I,level)、BW=im2bw(X,map,level)和 BW=im2bw(RGB,level)分别将灰度图像、索引色图像和真彩色图像 I 二值化为图像 BW。Level 是归一化的阈值，取值在[0,1]之间。

输入图像可以是 double 或 uint8 类型，输出图像为 uint8 类型。

例如对一幅图像进行二值化处理，结果如图 5.5 所示。

```
load trees
BW=im2bw(X,map,0.4)
imshow(X,map)
figure,imshow(BW)
```




图 5.5　将一幅索引色图像二值化的结果

说明： trees 是 MATLAB 预存的一个 mat 文件，包含一个名为 X 的数据矩阵，还有一个名为 map 的调色板。

3.　ind2gray

ind2gray 函数可以将索引色图像转换成灰度图像，其语法格式为：

```
I=ind2gray(X, MAP)
```

I=ind2gray(X,map)将具有调色板 map 的索引色图像 I 转换成灰度图像 I，它去掉了图像的色度和饱和度，仅保留了图像的亮度信息。输入图像可以是 double 或 uint8 类型，输出图像为 double 类型。

例如将一幅索引色图像转换成灰度图像，结果如图 5.6 所示。

```
load trees
I= ind2gray(X,map);
imshow(X,map)
```

```
figure,imshow(I)
```




图 5.6 将一幅索引色图像转化为灰度图像的结果

4. ind2rgb

ind2rgb 函数将索引色图像转换成真彩色图像，其语法格式为：

```
RGB=ind2rgb(X,map)
```

RGB=ind2rgb(X,map)将具有调色板 map 的索引色图像 X 转换成真彩色图像 RGB。实际实现时，就是产生一个三维数组，然后将索引色图像对应的调色板的颜色值赋予三维数组。输入图像 X 可以是 double 或 uint8 类型，输出图像 RGB 为 double 类型。

5. mat2gray

mat2gray 函数用于将一个数据矩阵转换成一幅灰度图像，其语法格式为：

```
I=mat2gray(A,[amin amax])
I=mat2gray(A)
```

这里 I=mat2gray(A,[amin amax])按指定的取值区间[amin amax]将数据矩阵 A 转换为灰度图像 I，amin 对应灰度 0(最暗)，amax 对应 1(最亮)。如果不指定区间[amin amax]，MATLAB 则自动将 A 阵中的最小元设为 amin，最大元设为 amax。

输入矩阵 A 和输出图像 I 都是 double 类型。实际上，mat2gray 函数与后面的 imshow 函数功能类似。imshow 函数也可以用来使数据矩阵可视化。

例如用 Sobel 算子对图像滤波，将滤波得到的数据矩阵转换为灰度图像，结果如图 5.7 所示。

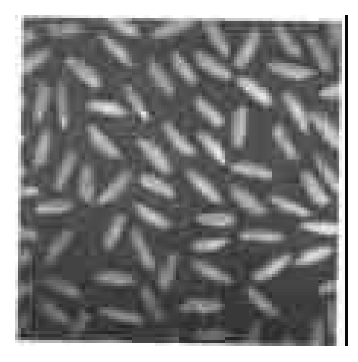
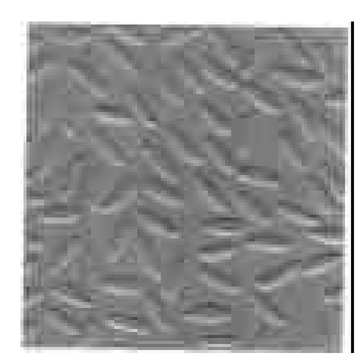



图 5.7 将数据矩阵表示为灰度图像

```
I=imread('rice.tif')
```

```
J=filter2(fspecial('sobel'),I);    %产生 Sobel 算子，并用 Sobel 算子对图像 I 进行滤波。
K= mat2gray(J);  %将数据矩阵转化成灰度图像。
imshow(I)
figure,imshow(K)
```

6. gray2ind

gray2ind 函数可以将灰度图像转换成索引色图像，其语法格式为：

```
[X,map]=gray2ind(I,n)
```

该函数按指定的灰度级数 n 和调色板 map，将灰度图像 I 转换成索引色图像 X，n 的默认值为 64。

7. grayslice

grayslice 函数通过设定阈值将灰度图像转换成索引色图像，其语法格式为：

```
X=grayslice(I,n)
X=grayslice(I,v)
```

这里 X=grayslice(I,n)将灰度图像 I 均匀量化为 n 个等级，然后转换为伪彩色图像 X。

X=grayslice(I,v)按指定的阈值向量 v(每一个元素都在 0 和 1 之间)对图像 I 的值域进行划分，而后转换成索引色图像 X。

输入图像 I 可以是 double 或 uint8 类型。如果阈值数量小于 256，则返回图像 X 的数据类型是 uint8，X 的值域为[0,n]或[0,length(V)]。否则，返回图像 X 为 double 类型，值域为[1,n+1]或[1,length(v)+1]。

例如将一幅灰度图像转换成索引色图像，结果如图 5.8 所示。

```
I=imread('ngc4024m.tif');
X=grayslice(I,16);        %设定灰度级为 16，将灰度图像转化为索引色图像。
imshow(I)
figure,imshow(x,hot(16))        %用 hot(16)产生调色板来显示索引色图像。
```

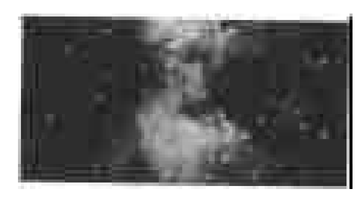



图 5.8 设定阈值将灰度图像转化为索引色图像

8. rgb2gray

rgb2gray 函数用于将一幅真彩色图像转换成灰度图像，其语法格式为：

```
I=rgb2gray(RGB)
newmap=rgb2gray(map)
```

其中 I=rgb2gray(RGB)命令将真彩色图像 RGB 转换成灰度图像 I，newmap=rgb2gray(map)将彩色调色板 map 转换成灰度调色板。

如果输入的是真彩色图像，则图像可以是 uint8 或 double 类型，输出图像 I 与输入图像类型相同。如果输入的是调色板，则输入和输出的图像都是 double 类型。

9. rgb2ind

rgb2ind 函数用于将真彩色图像转换成索引色图像，其语法格式为：

```
RGB=rgb2ind(X,map)
```

该命令将具有调色板 map 的索引色图像 X 转换成真彩色图像 X。输入的图像 X 可以是 double 或 uint8 类型，输出图像 RGB 为 double 类型。

5.2 颜色空间

在 MATLAB 图像处理工具箱中，总是直接(RGB 图像)或者间接(索引色图像)地使用 RGB 数据来表示颜色。但是除了 RGB 颜色模型之外，还有许多别的颜色模型，例如 HSV 模型，这些不同的颜色模型叫做色彩空间。

颜色模型是三维颜色空间中的一个可见光子集，它包括某个颜色域的所有颜色。常用的颜色模型有 NTSC、HSV 和 YCbCr 模型等。

NTSC 模型广泛应用于美国等国家的电视信号。它的特点是信号的强度信息和颜色信息相分离，同一个信号可以方便地同时表示彩色图像和黑白图像。在 NTSC 格式中，图像由 3 个分量表示：亮度(luminance)用 Y 表示；色度(hue)用 I 表示；饱和度(saturation)用 Q 表示。第一个分量亮度 Y，表示灰度信息，后两个分量表示彩色信息。因此 NTSC 模型使用的是 Y—I—Q 色彩坐标轴，NTSC 模型的色彩空间又称 YIQ 空间。

1. rgb2ntsc

rgb2ntsc 函数用于将 RGB 模型转换成 NTSC 模型，其语法格式为：

```
yiqmap=rgb2ntsc(rgbmap)
YIQ=rgb2ntsc(RGB)
```

其中 yiqmap=rgb2ntsc(rgbmap)将 RGB 空间中 m×3 的色彩表 rgbmap 转换成 YIQ 空间中的调色板 yiqmap。

YIQ=rgb2ntsc(RGB)将真彩色图像 RGB 转换为 YIQ 空间中的图像 YIQ。

2. ntsc2rgb

ntsc2rgb 函数用于将 NTSC 模型转换为 RGB 模型，其语法格式如下：

```
rgbmap=ntsc2rgb(yiqmap)
RGB=ntsc2rgb(YIQ)
```

rgbmap=ntsc2rgb(yiqmap)将 YIQ 空间中 m×3 的调色板 yiqmap 转换成 RGB 空间中的色彩表 rgbmap。RGB=ntsc2rgb(YIQ) 将 YIQ 色彩空间的图像 YIQ 转换为真彩色图像 RGB。

HSV 模型通常用于选择颜色，它是面向用户的一种复合主观感觉的色彩模型，比 RGB

模型更接近人们对颜色的感知。图 5.9 是 HSV 模型的示意图。

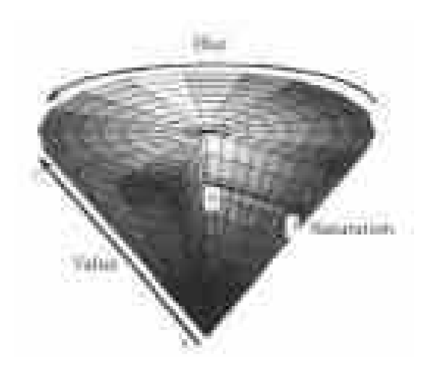


图 5.9 HSV彩色模型锥

HSV 模型中 H 表示颜色的种类，由绕 V 轴的旋转角决定，每一种颜色和它的补色之间相差 180°。S 为饱和度，V 为亮度。当色度由 0 变到 1，HSV 的颜色由红变为黄、绿、青、蓝、洋红，然后变回红色。当饱和度由 0 变到 1，颜色由不饱和变为饱和。当亮度由 0 变到 1，颜色越来越亮。

3. rgb2hsv

rgb2hsv 函数用于将 RGB 模型转换成 HSV 模型，其格式为：

```
hsvmap=rgb2hsv(rgbmap)
HSV=rgb2hsv(RGB)
```

其中 hsvmap=rgb2hsv(rgbmap)将 RGB 空间中 m×3 的色彩表 rgbmap 转换成 HSV 色彩空间中的调色板 hsvmap。HSV=rgb2hsv(RGB)将真彩色图像 RGB 转换为 HSV 空间中的图像 HSV。

4. hsv2rgb

hsv2rgb 函数用于将 HSV 模型转换为 RGB 模型，其格式为：

```
rgbmap=hsv2rgb(hsv,map)
RGB=hsv2rgb(HSV)
```

其中 rgbmap=hsv2rgb(hsv,map)将 HSV 色彩空间的调色板 hsvmap 转换为 RGB 空间中的色彩表 rgbmap，rgbmap 和 hsvmap 都是 m×3 的矩阵。RGB=hsv2rgb(HSV)将 HSV 色彩空间的图像 HSV 转换为真彩色图像 RGB。

YCbCr 模型广泛应用于数字视频。在 YCbCr 模型中，Y 为亮度，Cb 和 Cr 共同描述图像的色调，其中 Cb、Cr 分别为蓝色分量和红色分量相对于参考值的坐标。

5. rgb2Ycbcr

rgb2Ycbcr 函数用于将 RGB 模型转换成 YCbCr 模型，其语法格式为：

```
Ycbcrmap=rgb2Ycbcr(rgbmap)
YCBCR=rgb2Ycbcr(RGB)
```

这里 Ycbcrmap=rgb2Ycbcr(rgbmap)将 RGB 空间中的色彩表 rgbmap 转换成 YCbCr 空间

中的调色板 YCbCrmap。YCBCR=rgb2Ycbcr(RGB)将真彩色图像 RGB 转换为 YCbCr 空间中的图像 YCbCr。

6. ycbcr2rgb

ycbcr2rgb 函数用于将 YCbCr 模型转换成 RGB 模型，其语法格式为：

```
rgbmap=ycbcr2rgb(YCbCrmap)
RGB=ycbcr2rgb(YCBCR)
```

rgbmap=ycbcr2rgb(Ycbcrmap)将 YCbCr 空间中的调色板 YCbCrmap 转换成 RGB 空间中的色彩表 rgbmap。RGB=ycbcr2rgb(YCBCR)将 YCbCr 空间中的图像 YCbCr 转换为真彩色图像 RGB。

5.3 读写和显示图像文件

5.3.1 读写图像文件

MATLAB 为用户提供了专门的函数以从图像格式的文件中读写图像数据。这种方法不像其他编程语言一样，需要编写复杂的代码，只需要简单地调用 MATLAB 提供的函数即可。

MATLAB 支持的图像文件格式有*.cur、*.bmp、*.hdf、*.ico、*.jpg、*.pcx、*.png、*.tif 和*.xwd。用于图像文件 I/O 的工具箱函数是 imread、imfinfo 和 imwrite，下面一一介绍。

1. imread

imread 函数用于读入各种图像文件，其语法格式为：

```
A=imread(filename,fmt)
[X,map]=imread(filename,fmt)
[…]=imread(filename)
[…]=imread(…,idx)(只适用于*.cur、*.ico 和*.tif 格式)
[…]=imread(…,'backgroundcolor',BG)(只适用于*.png 文件)
[…]=imread(…,ref)(只适用于*.hdf 格式)
[A,map,alpha]=imread(只适用于*.png)
```

其中参数 fmt 指定了图像的格式，可选的值为 cur、bmp、hdf、ico、jpg、pcx、png、tif 和 xwd，图像格式也可以和文件名写在一起，即 filename.fmt，默认的文件目录为当前 MATLAB 的工作目录。如果不指定 fmt，MATLAB 会自动根据文件头确定文件格式。

[…]=imread(…,idx)用于读取多帧 TIFF 文件中的一帧。idx 为一个整数，表示图像的帧号。如果不指定 idx，则读取图像的第一帧。该命令也可以用于读取多帧 ico 或者 cur 文件中的一帧。其 idx 的含义和用法与 TIFF 格式的文件相同。

[…]=imread(…, ref)用于读取多帧 HDF 文件中的一帧。注意在 HDF 文件中，图像的帧号不一定与图像存储的序号一致。如果不指定 idx，则读取图像的第一帧。

读取 flowers.tif 的第 6 帧，可以从 MATLAB 工作空间观察读入变量的性质。

```
[X, map]=imread('flowers.tif',6);
```

```
whos X
  Name      Size          Bytes  Class
  X       362x500x3     543000  uint8 array
whos map
  Name      Size          Bytes  Class
  map       0x0              0  double array
```

2. imwrite

imwrite 函数用于输出图像，其语法格式为：

```
imwrite(A,filename,fmt)
imwrite(X,map,filename,fmt)
imwrite(…,filename)
imwrite(…,Parameter,Value,…)
```

与 imread 相似，参数 fmt 指定了图像的格式，可选的值是 cur、bmp、hdf、ico、jpg、pcx、png、tif 和 xwd，图像格式也可以和文件名写在一起，即 filename.fmt，默认文件目录为当前 MATLAB 的工作目录。

imwrite(X,map,filename,fmt)按照 fmt 指定的格式将图像矩阵 X 和调色板 map 写入文件 filename。如果 X 是 uint8 类型的，则 imwrite 把矩阵中的真实值写入文件；如果 X 是 double 类型的，imwrite 把矩阵值写入文件之前，先对其进行偏置，即写入的是 uint8(X-1)。

imwrite(…,Parameter,Value,…)可以让用户控制 HDF、JPEG、TIFF 3 种图像文件的输出。

例如 imwrite(X,map,'flowers.hdf','Compression','none',… 'WriteMode','append')

3. imfinfo

imfinfo 函数用于读取图像文件的有关信息，其语法格式为：

```
info=imfinfo(filename,fmt)
info=imfinfo(filename)
```

imfinfo 函数返回一个结构 info，它反映了该图像的各方面信息，其主要数据域包括：

- 文件名。如果该文件不在当前目录下，还包含该文件的完整路径。
- 文件格式。
- 文件格式的版本号。
- 文件的修改时间。
- 文件的大小。
- 文件的长度。
- 文件的宽度。
- 每个像素的位数。
- 图像类型，即该图像是真彩色(RGB)图像、索引色图像还是灰度图像。

用 imfinfo 可以详细地显示图像文件的各种属性，其属性的含义和值域可以参考 MATLAB 手册。

```
imfinfo('rice.tif')
ans =
Filename: [1x44 char]
```

```
FileModDate: [1x20 char]
FileSize: 65966
Format: 'tif'
FormatVersion: []
Width: 256
Height: 256
BitDepth: 8
ColorType: 'grayscale'
FormatSignature: [73 73 42 0]
ByteOrder: [1x13 char]
NewSubfileType: 0
BitsPerSample: 8
Compression: 'Uncompressed'
PhotometricInterpretation: 'BlackIsZero'
StripOffsets: [8x1 double]
SamplesPerPixel: 1
RowsPerStrip: 32
StripByteCounts: [8x1 double]
XResolution: 72
YResolution: 72
ResolutionUnit: 'Inch'
Colormap: []
PlanarConfiguration: 'Chunky'
TileWidth: []
TileLength: []
TileOffsets: []
TileByteCounts: []
Orientation: 1
FillOrder: 1
GrayResponseUnit: 0.0100
MaxSampleValue: 255
MinSampleValue: 0
Thresholding: 1
ImageDescription: [1x166 char]
```

实际上，在 MATLAB 中读取和存储图像还经常使用 load 和 save 这两个命令。load 的功能是从*.mat 文件中读取变量，save 命令的功能是将变量存入*.mat 文件。

save 命令的格式及意义如下：

```
save filename X              %将变量 X 存入文件名为 filename 的*.mat 文件。
save filename X Y Z          %把不同的变量存入同一个文件。
save filename X -ASCII       %以 ASCII 的格式把变量值存入*.mat 文件。
save filename X -ASCII-double    %以 16 位 ASCII 的格式存储变量。
save filename X append       %把变量 X 添加到文件名为 filename 的*.mat 文件中，原来存
                             储的变量还保持不变。
```

例如：把 flowers.tif 中的图像数据存入名为 newfile 的*.mat 文件中。

```
[I,map]=imread('flowers.tif');
save newfile I map
```

load 的格式为：

```
load filename              %把名为 filename 的*.mat 文件中存储的变量读入 MATLAB 的工作
                           空间，各变量名为存储时的变量名。
load filename X Y          %把名为 filename 的*.mat 文件中存储的 X 和 Y 的变量读入 MATLAB
                           的工作空间。
load filename.ext
load filename -ASCII
load filename -MAT
load('filename')
```

load filename.ext 可以读取名为 filename 的 ASCII 文件，文件可以存储各列用空格分开的 ASCII 格式的变量值，文件中还可以包含注释符号“%”。load filename –ASCII 和 load filename –MAT 决定是以 ASCII 格式还是以二进制格式读取变量。

当用字符串格式表示文件名时，load('filename')可以直接读取名为 filename 的*.mat 文件。

例如读取刚才用 save 存储的 newfile.mat 中的变量 I：

```
load newfile I
whos
  Name       Size          Bytes  Class
  I        362x500x3      543000   uint8    array
  ans         1x1          5184    struct   array
  map         0x0            0     double   array
```

5.3.2　图像文件的显示

MATLAB 及图像处理工具箱的显示功能非常强大，不仅可以用来显示各种类型的图像，还可以用多种方式显示图像及图像序列。下面对这些函数进行介绍。

1.　image

image 是 MATLAB 本身提供的最原始的图像显示函数，其使用格式如下：

```
image(X);
colormap(map);
```

为了用 image 函数显示由矩阵表示的图像，MATLAB 将矩阵的每个元素对应到当前调色板的一行，并取这一行的颜色值作为该点的颜色。因此显示图像时必须先指定调色板，才能真实地表示图像。

例如：MATLAB 预存的一组图像数据，包含图像矩阵 X 和调色板矩阵 map，下面的语句用来显示图，其图像如图 5.10 所示。

```
load clown
image(X)
colormap(map)
axis equal
axis('off')
```



图 5.10 用image函数显示图像

说明： clown 也是 MATLAB 预存的一个 mat 文件，里面包含一个数据矩阵 X 和一个调色板 map。

另一个与 image 函数相似的函数是 imagesc，两者的区别在于 imagesc 能够自动调整值域范围。

2. imshow

imshow 函数是最常用的显示各种图像的函数，其语法如下：

```
imshow(I,n)
imshow(I,[low high])
imshow(BW)
imshow(X,map)
imshow(RGB)
imshow(…,display_option)
imshow(x,y,A,…)
imshow filename
```

imshow(I,n)和 imshow(I,[low high])用于显示灰度图像，n 为灰度级数目，默认值为 256。[low high]为图像数据的值域。在很多情况下，经过处理的图像数据的值域都会发生变化。比如对一幅 double 型的灰度图像滤波后，图像数据的值域已不在[0,1]中了，如果还用前面的显示方法，则得不到正确的结果。如果清楚地知道数据的值域[low high]，可以使用调用 imshow(I,[low high])。否则可用空向量为参数，即 imshow(I,[])。

imshow(BW)用于显示二值图像，图像 I 的数据类型可以是 double 或 uint8，值域为 0 或 1。imshow(X,map)用于显示索引色图像，X 为数据图像矩阵，map 为调色板。imshow(RGB)用于显示真彩色图像，图像 I 的数据类型可以是 double，值域为[0，1];也可以是 uint8，值域为[0，255]。imshow filename 可以直接显示图像文件，不过图像数据并没有进入 MATLAB 的工作空间，若要处理图像，则用 imread 读入数据，或用 getimage 从句柄对象中得到数据。

下面的语句用来输出一幅图像滤波后的结果，如图 5.11 所示。

```
I=imread('rice.tif')
J=filter2([1 2;-1 -2],I)
%用模板[1 2;-1 -2]对图像滤波。
imshow(I)
figure,Imshow(J,[])     %由于滤波后图像灰度范围与滤波之前不同，所以用[]来作为参数。
```

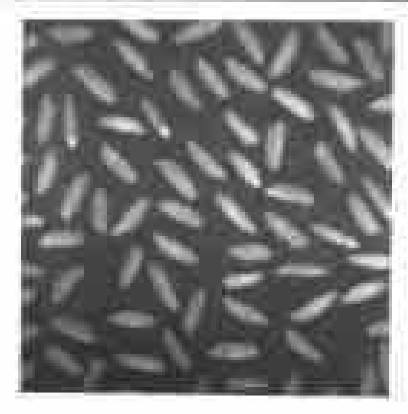



图 5.11 rice.tif滤波之后的图像

3. colorbar

colorbar 函数用于显示颜色条，并将图像中使用到的色彩排列在图像旁边，这样可以根据图像的色彩确定各像素的值。这对于用图像表示灰度范围不在通常范围内的情况非常有用。

colorbar 函数语法格式如下：

```
colorbar('vert')
colorbar('horiz')
colorbar(h)
colorbar
h=colorbar(…)
```

其中 colorbar('vert')、colorbar('horiz')分别指定了颜色条的显示方式为垂直或水平，默认值为垂直('vert')。colorbar(h)将颜色条放在指定的坐标轴 h 上，h 为句柄。h=colorbar(…)返回颜色条坐标轴的句柄。

下面的例子用 LoG 算子对图像滤波，在输出图像上加颜色条，其结果如图 5.12 所示。

```
I=imread('saturn.tif');
H=fspecial('log');          %产生拉普拉斯高斯算子。
I2=filter2(h, I)            %用 LoG 算子对图像进行滤波。
Imshow(I2, []),
colormap(jet(64)),
colorbar                    %将颜色条放置在坐标轴上。
```

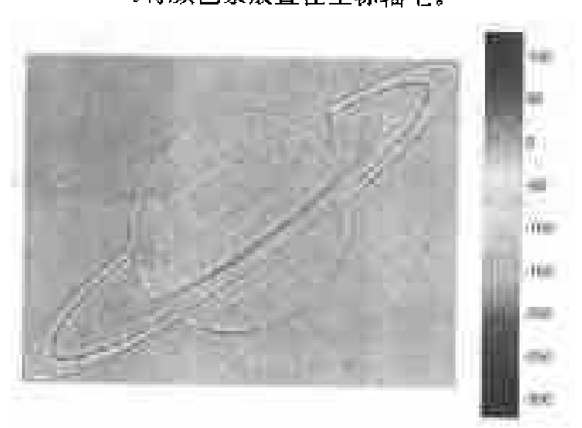


图 5.12 为灰度图像滤波后的结果加颜色条

4. montage

多帧图像指的是包含不止一幅图像的图像，MATLAB 中支持多帧图像的文件格式有 HDF 和 TIFF 两种。多帧图像的显示方式也有两种，一种是显示多帧图像中的一帧，另一种是同时显示多帧图像的所有帧。

前面提到，MATLAB 中多帧图像是用四维数组表示的，因此为了显示多帧图像中的一帧，可以指定图像的帧号。例如读取多帧图像 mri 的第 3 帧，我们可以用下列语句实现，其结果如图 5.13 所示。

```
mri=uint8(zeros(128,128,1,27));
for frm=1:27
    [mri(:,:,:,frm),map]=imread('mri.tif',frm);
end
imshow(mri(:,:,:,3),map)
```

另外还可以同时显示多帧图像中的所有帧，这是由 montage 函数完成的，其语法格式为：

```
montage(I)                %拼接灰度图像。
montage(X,map)            %拼接索引色图像。
montage(RGB)              %拼接真彩色图像。
h=montage(…)              %返回拼接后的图像句柄。
```

下面的例子将序列图像拼接显示，其结果如图 5.14 所示。

```
mri=uint8(zeros(128,128,1,27));
for frm=1:27
    [mri(:,:,:,frm),map]=imread('mri.tif',frm);
end
montage(mri,map)
```

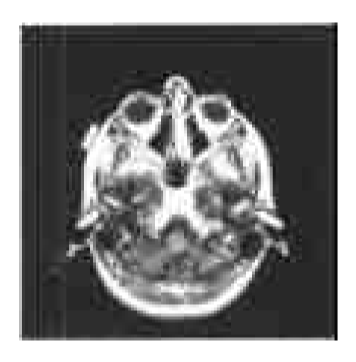


图 5.13 mri图像中的第3帧

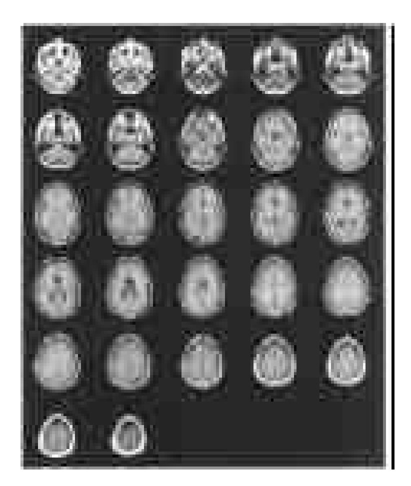


图 5.14 同时显示多帧图像

5. immovie

在 MATLAB 中，我们不但可以单独显示多帧图像中的各帧，同时显示所有帧，还可以用动画的方式显示帧，immovie 函数即可以将多帧图像转换成 MATLAB 动画，其语法格

式为：

```
mov=immovie(X,map)
```

immovie 只能使用索引色图像，因此如果要将其他类型的图像转换成动画，首先要转换成索引色图像。要在 MATLAB 中播放这个动画，首先要指定调色板。

一般可以用下例语句放映动画：

```
colormap(map), movie(mov)
```

例如：

```
mri=uint8(zeros(128,128,1,27));
for frm=1:27
    [mri(:,:,:,frm),map]=imread('mri.tif',frm);
end
mov=immovie(mri,map);
colormap(map)
movie(mov)
```

6.　subimage

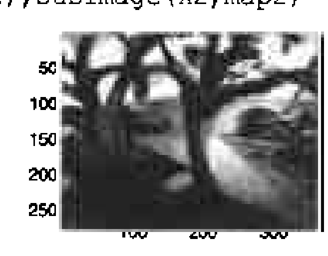
MATLAB 提供的 subplot 函数虽然能将一个图像窗口分成几个部分，但同一个图像窗口内只能有一个调色板。而 MATLAB 图像处理工具箱的 subimage 函数可在一个图像窗口内使用多个调色板，使得各种图像能在同一个图像窗口中显示。

subimage 函数的语法格式为：

```
subimage(X,map)        %在一个窗口里显示多个索引色图像。
subimage(I)            %在一个窗口里显示多个灰度图像。
subimage(RGB)          %在一个窗口里显示多个真彩色图像。
subimage(x,y,…)        %将图像按指定的坐标系(x,y)显示。
h=subimage(…)          %返回图像对象的句柄，其中输入的图像可以是 uint8 或 double 类型。
```

下面的例子将显示两幅具有不同调色板的图像，结果如图 5.15 所示。

```
load trees
[X2, map2]=imread('forest.tif');
subplot(1,2,1),subimage(X,map)
subplot(1,2,2),subimage(X2,map2)
```



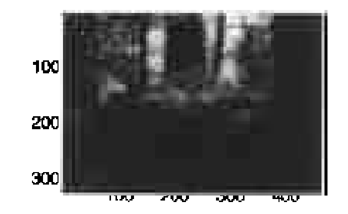


图 5.15　在同一图形窗口显示两幅图像

7.　zoom

有时图像的大小不适合观察，这时就需要对图像进行缩放，MATLAB 提供的 zoom 函

数用于缩放图像，其语法格式很多，主要有下面几种：

```
zoom on
zoom off              %关闭缩放功能。
zoom out              %恢复图像的原始尺寸。
zoom reset            %将当前图像尺寸设为缩放的起点。
zoom xon              %设置 x 轴方向的缩放功能。
zoom yon              %设置 y 轴方向的缩放功能。
zoom (factor)         %按照指定的缩放系数 factor 进行缩放。
zoom (fig,option)     %对指定的图像 fig 进行缩放，option 就是上述各选项。
```

其中 zoom on 命令使用户可用鼠标指针缩放图像。用户可用鼠标指针选择一个图像点或拉出一个矩阵框，单击左键将放大图像，单击右键将缩小图像。

例如我们要对 ic.tif 图像进行缩放，首先读入 ic.tif 图像进行显示，然后用鼠标指针在图像上选择矩形区域，自动对选择区域进行放大，结果如图 5.16 所示。然后单击鼠标右键则恢复为原来大小。

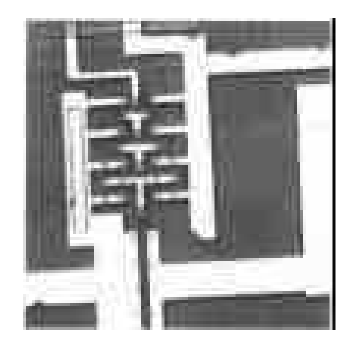
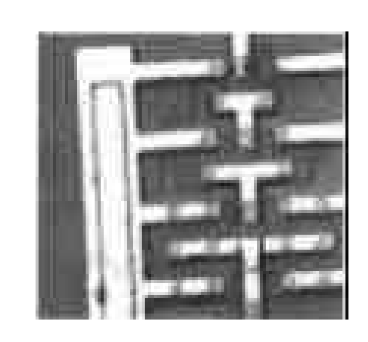



图 5.16 用zoom函数对图像放大的结果

8. warp

调用 imshow 函数时，默认情况下 MATLAB 将图像显示在二维平面上。但是也可以将图像显示在别的类型的表面上，比如球面。通过插值把图像显示在特定的表面上称为纹理映射。warp 函数用于显示图像的纹理映射，其语法格式为：

```
warp(X,map)
warp(I,n)
warp(RGB)
warp(z,…)
warp(x,y,z,…)
h=warp(…)
```

warp 函数可用纹理映射的方法将一幅图像显示在一个参数曲面上。warp(X,map)、warp(I,n)、warp(RGB)分别将索引色图像、灰度图像和真彩色图像映射到显示矩形平面区域上。warp(z,…)将图像映射到曲面上，warp(x,y,z…)将图像映射到曲面(x,y,z)上。h=warp(…)返回映射后到图像类型。

下面的例子将一幅图像映射到柱面上显示，结果如图 5.17 所示。

```
[x,y,z]=cylinder;
I=imread('testpat1.tif');
warp(x,y,z,I);
```

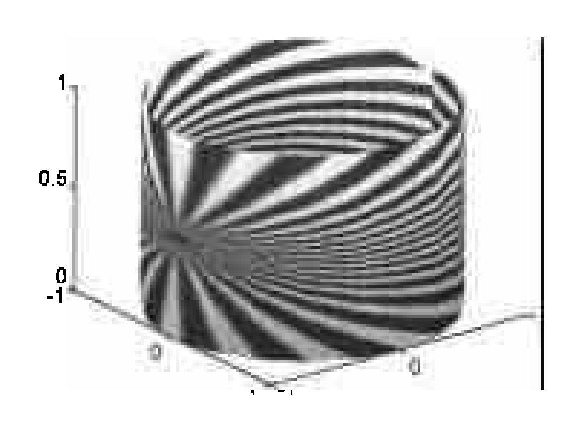


图 5.17　显示图像纹理

5.4　图像的几何操作

在处理图像的过程中，有时需要对图像的大小和几何关系进行调整，比如对图像进行缩放及旋转，这时图像中每个像素的值都要发生变化。数字图像的坐标是整数，经过这些变换之后的坐标不一定是整数，因此要对变换之后的整数坐标位置的像素值进行估计。MATLAB 提供了一些函数实现这些功能。

5.4.1　图像的插值

插值是常用的数学运算，通常是利用曲线拟合的方法，通过离散的采样点建立一个连续函数来逼近真实曲线，用这个重建的函数便可求出任意位置的函数值。

设已知函数值为 w1,w2,...，则未知点 x 的函数值通过插值可以表示为：

$$f(x)=\sum_{l=1}^{L}w_l h(x-x_l)$$

其中 h(•)为插值核函数，w_l 为权系数。

插值算法的数值精度及计算量与插值核函数有关，插值核函数的设计是插值算法的核心。MATLAB 图像处理工具箱提供了 3 种插值方法：最近邻插值(nearest neighbor interpolation)、双线性插值(bilinear interpolation)和双三次插值(bicubic interpolation)。

1.　最近邻插值

最近邻插值是最简单的插值，在这种算法中，每一个插值输出像素的值就是在输入图像中与其最临近的采样点的值。该算法的数学表示为：

$$f(x)=f(x_k)，如果\ 1/2(x_{k-1}+x_k)<x<1/2(x_k+x_k+1)$$

最近邻插值是工具箱函数默认使用的插值方法，而且这种插值方法的运算量非常小。对于索引色图像来讲，它是唯一可行的方法。不过，最近邻插值法的值核频域特性不好，从它的傅立叶谱上可以看出，它与理想低通滤波器的性质相差较大。当图像含有精细内容，也就是高频分量时，用这种方法实现倍数放大处理，在图像中可以明显看出块状效应。

2. 双线性插值

双线性插值法的输出像素值是它在输入图像中2×2领域采样点的平均值，它根据某像素周围4个像素的灰度值在水平和垂直两个方向上对其插值。

设 $m<i'<m+1, n<j'<n+1, a=i'-m, b=j'-n$，i'和j'是要插值点的坐标，则双线性插值的公式为：

$$g(i',j')=(1-a)(1-b)g(m,n)+a(1-b)g(m+1,n) \\ +(1-2)bg(m,n)+abg(m+1,n+1)$$

把按照上式计算出来的值赋予图像的几何变换对应于(i',j')处的像素，即可实现双线性插值。

3. 双三次插值

双三次插值的插值核为三次函数，其插值邻域的大小为4×4。它的插值效果比较好，但相应的计算量也较大。

5.4.2 图像的插值缩放和插值旋转

MATLAB 图像处理工具箱中的函数 imresize 可以用上述 3 种方法对图像进行插值缩放，如果不指定插值方法，则默认使用最近邻插值法。

1. imresize

imresize 函数的语法格式为：

```
B=imresize(A,m,method)
B=imresize(A,[mrows ncols],method)
B=imresize(…,method,n)
B=imresize(…,method,h)
```

这里参数method用于指定插值的方法，可选的值为 nearest(最近邻法)、bilinear(双线形插值)及 bicubic(双三次插值)，默认值为 nearest。

B=imresize(A,m,method)返回原图 A 的 m 倍放大图像(m 小于 1 时效果是缩小)。

B=imresize(A,[mrows ncols],method)返回一个 mrows 行、ncols 列的图像，若 mrows 和 ncols 定义的长宽比与原图不同，则图像会产生变形。

在使用 bilin 和 bicubic 方法缩小图像时，为消除引入的高频成分，imresize 使用一个前端平滑滤波器，默认的滤波器尺寸为 11×11。用户也可通过参数 n 指定滤波器的尺寸，即 B=imresize(…,method,n)。对于 nearest 插值方法，imreasize 不使用前端滤波器，除非函数明确指定。

B=imresize(…,method,h)使用用户设计的插值核 h 进行插值，h 可以看作一个二维 FIR 滤波器。

图 5.18 是使用不同插值方法对放大图像结果进行比较的效果图，其命令格式如下：

```
load woman2
```

```
imshow(X,map);
X1=imresize(X,2,'nearest');
figure,imshow(X1,[]);
X2=imresize(X,2,'bilinear');
figure,imshow(X2,[]);
X3=imresize(X,2,'bicubic');
figure,imshow(X3,[]);
```




图 5.18　3种不同插值方法对图像进行放大的结果

从 3 种插值方法对图像放大的结果可以看出在进行小倍数放大时，最近邻插值方的效果还可以，双线性插值方法的结果有些模糊，双三次插值效果最好。

2.　imrotate

在对数字图像进行旋转的时候，各像素的坐标将会发生变化，使得旋转之后不能正好落在整数坐标处，需要进行插值。在工具箱中的函数 imrotate 可用上述 3 种方法对图像进行插值旋转，默认的插值方法也是最近邻插值法。

imrotate 的语法格式为：

```
B=imrotate(A,angle,method)
b=imrotate(A,angle,method,'crop')
```

函数 imrotate 对图像进行旋转，参数 method 用于指定插值的方法，可选的值为 nearest(最近邻法)、bilinear(双线形插值)及 bicubic(双三次插值)，默认值为 nearest。一般来说，旋转后的图像会比原图大，超出原图像的部分值为 0。用户也可以指定 crop 参数对旋转后的图像进行剪切(取图像的中间部分)，使返回的图像与原图大小相同。

下面的例子可将图像插值旋转 35°，结果如图 5.19 所示。

```
I=imread('ic.tif');
J=imrotate(I,35,'bilinear');
imshow(I)
```

```
figure,imshow(J)
```

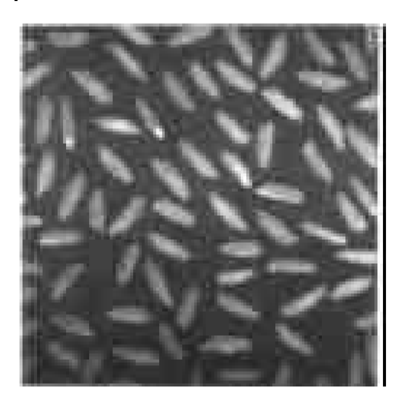
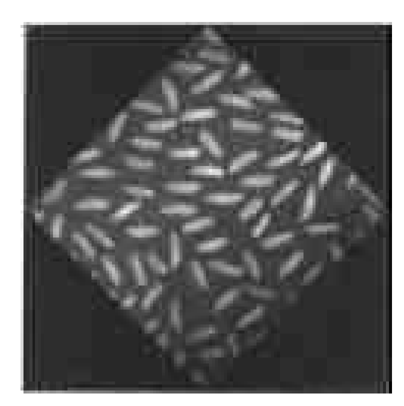



图 5.19 对图像进行插值旋转的结果

5.4.3 图像的剪切

有时只需要处理图像中的一部分，或者需要将某一部分取出，这样就要对图像进行剪切。工具箱中的函数 imcrop，用于剪切图像中的一个矩形子图，用户可以通过参数指定这个矩形顶点的坐标，也可以用鼠标指针选取这个矩形。

imcrop 函数的语法格式为:

```
I2=imcrop(I)
X2=imcrop(X,map)
RGB2=imcrop(RGB)
I2=imcrop(I,rect)
X2=imcrop(X,map,rect)
RGB2=imcrop(RGB,rect)
[…]=imcrop(x,y,…)
[A,rect]=imcrop(…)
[x,y,A,rect]=imcrop(…)
```

其中 I2=imcrop(I)、X2=imcrop(X，map)和 RGB2=imcrop(RGB)为交互式地对灰度图像、索引色图像和真彩色图像进行剪切。I2=imcrop(I,rect)、X2=imcrop(X,map,rect)和 RGB2=imcrop(RGB,rect)按指定的矩形框 rect 剪切图像，rect 是一个四元向量[xmin ymin width height]，分别表示矩形的左下角的坐标和长度及宽度。[…]=imcrop(x,y,…)在指定坐标系(x，y)中剪切图像。[A,rect]=imcrop(…)和[x,y,A,rect]=imcrop(…)在用户交互剪切图像的同时返回剪切框的参数 rect。

下面的例子将从一幅图像中剪切一块子图，坐标为(60,40)～(100,90)，结果如图 5.20 所示。

```
I=imread('ic.tif');
I2=imcrop(I,[60 40 100 90]);    %对读入的图像指定切割范围[60 40 100 90]。
imshow(I)
figure,imshow(I2)
```

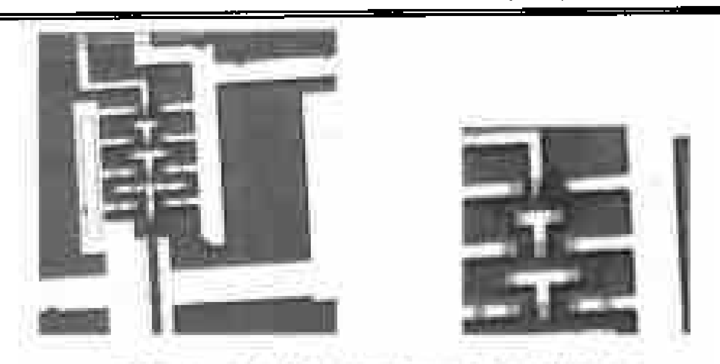


图 5.20　从图像中剪切出一块子图像的结果

5.5　图像邻域和块操作

5.5.1　滑动邻域操作

在 MATLAB 中，滑动邻域是一个像素集，像素集包含的元素由中心像素的位置决定。滑动邻域操作一次只处理一个图像像素。当操作从图像矩阵的一个位置移动到另一个位置时，滑动邻域也以相同的方向运动，其示意图如图 5.21 所示。

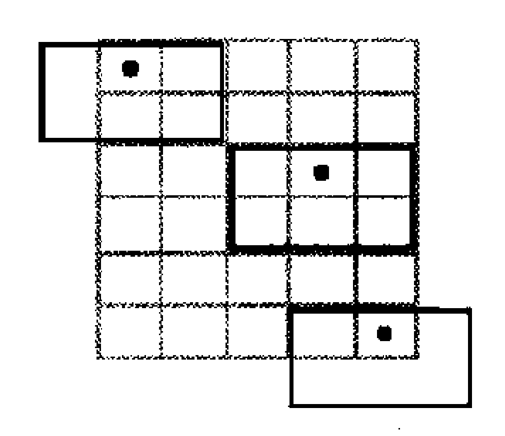


图 5.21　滑动邻域示意图

上图的滑动邻域是一个 2×3 的矩阵，黑点表示中心像素。对于 m×n 的滑动邻域来说，中心像素的位置是：

```
floor(([m,n]+1)/2)
```

上图中 2×3 滑动邻域的中心就是(1，2)。

在 MATLAB 中进行滑动邻域操作的具体步骤如下：

(1)　选择像素。

(2)　确定该像素的滑动邻域。

(3)　调用合适的函数对滑动邻域中的元素进行计算。

(4)　将计算结构作为输出图像中对应的像素的值。

(5)　重复计算，遍及图像中的所有像素。

MATLAB 提供的邻域操作函数有以下几种：

1. colfilt

MATLAB 图像处理工具箱中的 colfilt 函数用于快速地邻域操作，其语法格式如下：

```
B=colfilt(A,[m n],block_type,fun)
B=colfilt(A,[m n],block_type,fun,P1,P2,…)
B=colfilt(A,[m n],[mblock nblock] ,block_type,fun,…)
B=colfilt(A,'indexed',…)
```

说明： B=colflt(A,[m n],block_type，fun)实现快速的邻域操作，图像块的尺寸为 m×n，block_type 指定了块的移动方式，即：

- block_type='distinct'，图像块不重叠。
- block_type='sliding'，图像块滑动。

fun 为运算函数，其形式为 y=fun(x)。B=colilt(A,[m n],block_type,fun,P1,P2,…)指定 fun 中除 x 以外的其他参数 P1、P2、…。B=colflt(A,[m n],[mblock nblock],block_type,fun,…)为节省内存按 mblock×nblock 的图像块对图像 A 进行块操作。

下面的例子将对图像进行滑动平均，结果如图 5.22 所示。

```
I=imread('alumgrns.tif');
I2=colfilt(I,[5 5],'sliding','mean');
imshow(I)
figure,imshow(I2, []);
```

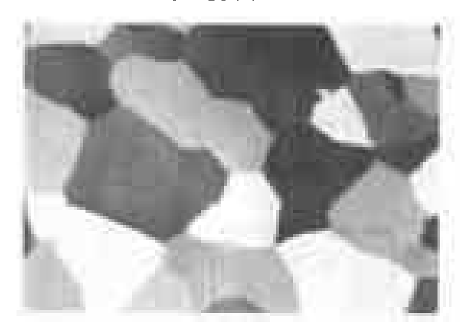



图 5.22 对图像进行滑动平均的结果

对于滑动邻域操作，colfilt 函数为图像中每个像素建立一个列向量，向量的各元素对应该像素的邻域的元素。下图显示了一个 6×5 的图像按照 2×3 的邻域进行处理的情况。colfilt 函数为图像建立了一个 30 列的矩阵，每列有 6 个元素。colfilt 函数可以根据需要对图像进行补零。例如图 5.23 中的图像右角的像素有两个 0 元素，这是因为对图像补零的结果。

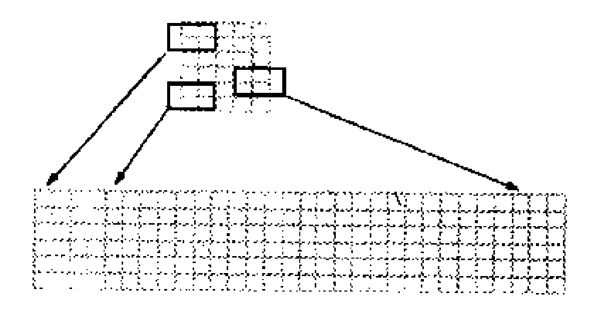


图 5.23 滑动邻域操作生成的临时矩阵

colfilt 函数生成的临时矩阵被传递给自定义函数，自定义函数为矩阵的每一列返回一个单独的值。MATLAB 中很多函数有这种功能，比如 mean、std 等。返回值赋给输出图像中对应的像素。

下面的例子是对输入图像处理，输出图像为每个像素邻域的最大值。

```
f=inline('max(x)');
J=colfilt(I,[8 8],'sliding',f);
```

注意： 这里 f 的定义是 max(x)，而不是 max(x(:))。这是因为在原图像中，每个像素的邻域已经被排列成列向量。

colfilt 实际上也可以进行下面提到的图像块操作。对于图像块操作，colfilt 函数把每个图像块排列成一列，构成一个临时矩阵。如果需要，可以对图像进行补零。

图 5.24 显示了一个 6×16 的图像按照 4×6 的图像块进行处理的情况。图像首先被补零至 8×16 大小，构成 6 个 4×6 的图像块。然后每个图像块被排列成一列，形成了 24×6 的临时矩阵。

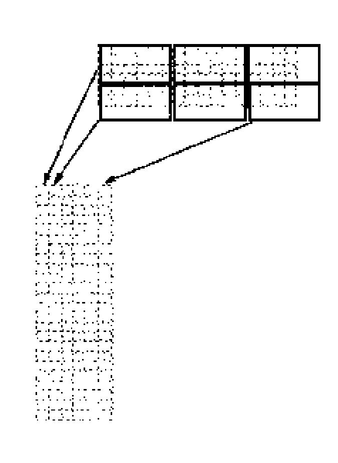
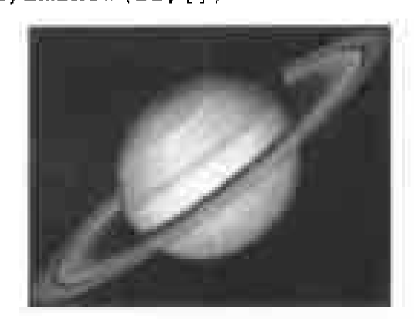


图 5.24 图像块操作生成的临时矩阵

colfilt 函数把原始图像排列成临时矩阵之后，将其传入自定义函数。自定义函数必须返回和临时矩阵大小相同的矩阵。然后 colfilt 函数再把结果重新排列成原始图像的格式。

下面的例子利用 colfilt 函数把输入图像的 8×8 的图像块的均值赋予图像块中所有元素，得到的结果如图 5.25 所示。

```
I=imread('saturn.tif');
imshow(I)
f=inline('ones(64,1)*mean(x)');
I2=colfilt(I,[8 8],'distinct',f);
figure,imshow(I2,[])
```



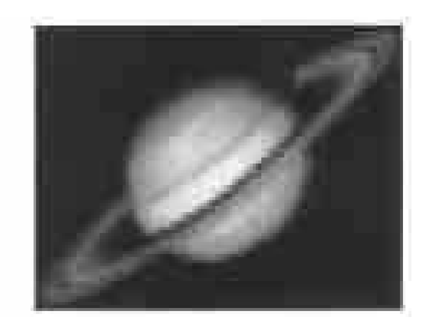


图 5.25 图像块操作求平均的结果

2. nlfilter

nlfilter 函数是通用的滑动窗操作函数，其语法格式为：

```
B=nlfilter(A,[m n],fun)
```

```
B=nlfilter(A,[m n],fun,P1,P2,…)
B=nlfilter(A,'indexed',…)
```

这里 B=nlfilter(A,[m n],fun)是通用的滑动窗操作，图像块的尺寸为 m×n。

fun 为运算函数，其形式为 y=fun(x)。B=nlfilter(A,[m n],fun,P1,P2,…)指定 fun 中除 x 以外的其他参数 P1、P2、……。B=nlfilter(A,'indexed',…)是对索引色图像的滑动窗操作。

5.5.2 图像块操作

图像块操作是将图像的数据矩阵划分为同样大小的矩形区域的操作，它是图像分析和图像压缩的基础。同时由于图像划分为图像块后可以转化为矩阵或者向量运算，因此可以大大加快图像处理的速度。

图像块的定义将图像的数据矩阵划分为同样大小的矩形区域，不同的图像块在图像上面排列，相互之间没有重叠。它的排列顺序是从左上角开始。如果图像不能恰好被划分，则在图像的右下部对其补零。如图 5.26 所示，一个 15×30 的图像块按照 4×8 的图像块进行划分，结果在图像的右边补了两列 0，在图像的下边补了一行 0。

图 5.26 图像块的划分

MATLAB 提供的滑动窗操作函数有以下几种：

1. bestblk

bestblk 函数用于选择图像块的尺寸，其语法格式为：

```
siz=bestblk([m n],k)
[mb,nb]=bestblk([m n],k)
```

其中 siz=bestblk([m n],k)返回对尺寸为 m×n 的图像的块划分 siz，k 为图像块长度和宽度的最大值。[mb,nb]=bestblk([m n],k) 返回 mb 和 nb 分别为图像块的行数和列数。

举例如下：

```
siz=bestblk([768 1024], 56)
siz =
   48    32
```

2. blkproc

blkproc 是通用的图像块操作函数，其语法格式为：

```
B=blkproc(A,[m n],fun)
B=blkproc(A,[m n],fun,P1,P2,…)
B=blkproc(A,[m n],[mborder nborder],fun,…)
B=blkproc(A,'indexed',…)
```

其中 B=blkproc(A,[m n],fun)按 m×n 的图像块划分对图像 A 做运算，fun 为运算函数，其形式为 y=fun(x)。B=blkproc(A,[m n],fun,P1,P2,…)指定 fun 中除 x 以外的其他参数 P1、P2、……。B=blkproc(A,[m n],[mborder nborder],fun,…)指定图像块的扩展边界 mborder 和 nborder，实际图像块大小为(m+2*mborder)*(n+2*nborder)。

下面的例子为计算图像 8×8 区域的局部标准差，结果如图 5.27 所示。

```
I=imread('tire.tif');
f=inline('uint8(round(std2(x)*ones(size(x))))');
I2=blkproc(I,[8 8],f);
imshow(I)
figure,imshow(I2,[]);
```

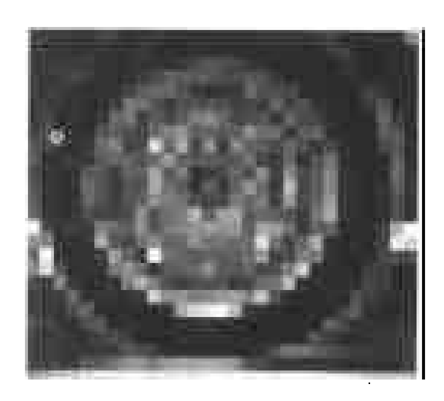



图 5.27 计算图像块局部标准差的结果

说明： 图像块的标准差计算公式为 std2(x)，结果为标量。但是为了使运算之后的图像与原图像大小相同，因此乘以一个与图像块大小相同的方阵 ones(size(x))，指定的函数就变为 std2(x)*ones(size(x))。blkproc 函数并不需要结果图像与原图像大小相同，但是如果希望结果图像与原图像大小相同，就必须定义合适的变换函数。

B=blkproc(A,[m n],[mborder nborder],fun,…)允许进行图像块操作时，各图像块之间有重叠。也就是说，在对每个图像块进行操作时，可以为图像块增加额外的行和列。当图像块有重叠时，blkproc 函数把扩展的图像块传递给自定义的函数。

图 5.28 显示了一个 15×30 的图像块，各块之间有 1×2 的重叠的情况。每个 4×8 的块，在它的上面和下面各重叠了一行，在左边和右边各重叠了一列，图像中重叠部分用阴影表示。

为了实现上面的重叠方法，可以用如下语句：

```
B=blkproc(A,[4 8],[1 2],'fun')
```

允许各图像块之间重叠通常会给图像补零。比如上面的 15×30 的图像块，允许重叠之后就变成了 15×30 的图像。

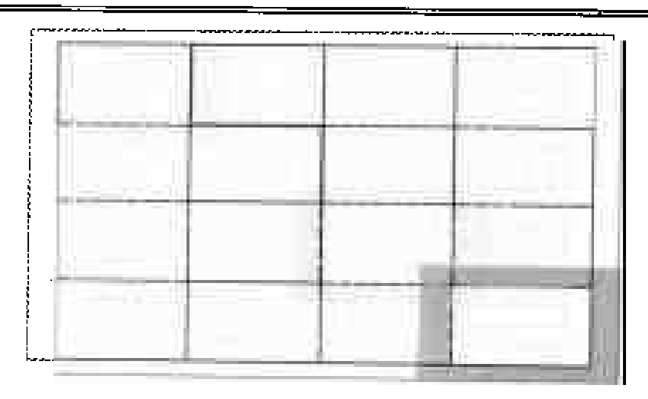


图 5.28 图像块的重叠划分

3. col2im

col2im 函数用于将向量重新排列成图像块。将向量重新排列成图像块的好处在于，MATLAB 的向量计算功能特别强大，如果将矩阵转化为向量，则使速度提高很多。但是处理完之后还要将向量排列成矩阵。

col2im 语法格式为：

```
A=col2im(B,[m n],[mm nn],block_type)
```

这里 A=col2im(B,[m n],[mm nn],block_type)将图像 B 的每一列重新排列成 m×n 的图像块，block_type 指定了排列的方式，即：

- block_type='distinct'，图像块不重叠。
- block_type='sliding'，图像块滑动。

用这些图像块组合成 mm×nn 的图像 A。

4. im2col

im2col 函数实现将图像块排列成向量的功能，其语法格式为：

```
B=im2col(A,[m n],block_type)
B=im2col(A,[m n])
B=im2col(A,'indexed',…)
```

这里 B=im2col(A,[m n],block_type)将图像 A 的每一个 m×n 块转换成一列，重新组合成图像 B。

block_type 指定了图像块的排列方式，即：

- block_type='distinct'，图像块不重叠。
- block_type='sliding'，图像块滑动。

5.6 特定区域处理

5.6.1 指定区域

在进行图像处理时，有时只要对图像中的某个特定区域进行处理，并不需要对整个图

像进行处理，比如要对用户选定的一个区域作均值滤波或对比度增强，MATLAB 就可以只对指定的区域进行处理。

MATLAB 中对特定区域的处理是通过二值掩模来实现的。用户选定一个区域后会生成一个与原图大小相同的二值图像，选定的区域为白色，其余部分为黑色。通过掩模图像，就可实现对特定区域的选择性处理。

MATLAB 图像处理工具箱提供了 4 个函数分别支持对特定区域的选择、滤波和填充，下面分别给以介绍。

1. roipoly

roipoly 函数用于选择图像中的多边形区域，其语法格式如下：

```
BW=roipoly(I,c,r)
BW=roipoly(I)
BW=roipoly(x,y,I,xi,yi)
[BW,xi,yi]=roipoly(…)
[x,y,BW,xi,yi]=roipoly(…)
```

roipoly 函数返回二值图像 BW，选中的区域值为 1，其余的部分值为 0。这个二值图像可以作为掩模，通过与原图的运算选择目标或背景。

BW=roipoly(I,c,r)是用向量 c、r 指定多边形各角点的 X、Y 轴的坐标。

BW=roipoly(I)是让用户交互选择多边形区域，选择角点，用空格键和 Del 键撤销选择，按 Enter 键确认选择。

BW=roipoly(x,y,I,xi,yi)用矢量 x 和 y 建立非默认的坐标系，然后在指定的坐标系下选择由向量 xi、yi 指定的多边形区域。[BW,xi,yi]=roipoly(…)交互选择多边形区域，并返回多边形角点的坐标。[x,y,BW,xi,yi]=roipoly(…)交互选择多边形区域后，还返回多边形顶点在指定的坐标系 X－Y 下的坐标。

下面的例子将根据指定的坐标选择一个六边形区域，结果如图 5.29 所示。

```
I = imread('eight.tif');
c = [222 272 300 270 221 194];
r = [21 21 75 121 121 75];
BW = roipoly(I,c,r);
imshow(I)
figure, imshow(BW)
```

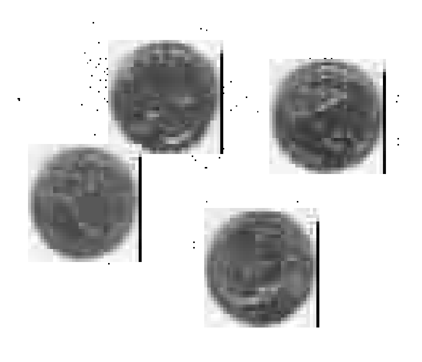
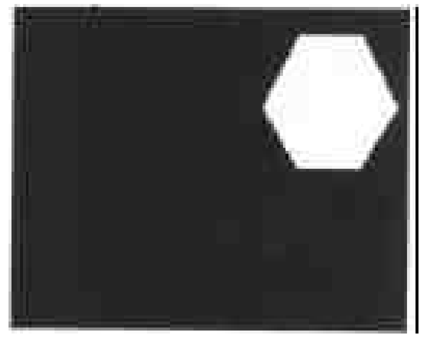



图 5.29 根据指定的坐标选择六边形

2. roicolor

另外 MATLAB 图像处理工具箱提供的 roicolor 函数可以实现按灰度选择区域，其语法格式为：

```
BW=roicolor(A,low,high)
BW=roicolor(A,v)
```

这里 BW=roicolor(A,low,high)按指定的灰度范围分割图像，返回二值掩模 BW，[low high]为所要选择区域的灰度范围。如果 low 大于 high，则返回为空矩阵。BW=roicolor(A,v)是按向量 v 中指定的灰度值来选择区域。

下面的例子是按灰度分割图像中的目标，结果如图 5.30 所示。

```
I=imread('rice.tif')
BW=roicolor(I,128,255)      %选择图像灰度范围在 128 和 255 之间的像素。
imshow(I)
figure,imshow(BW)
```

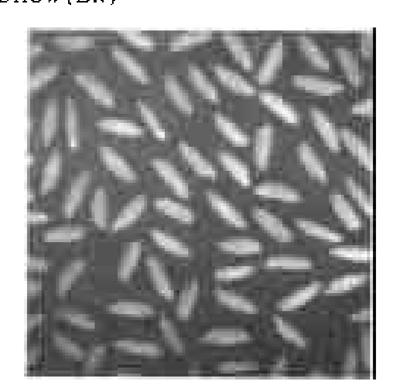
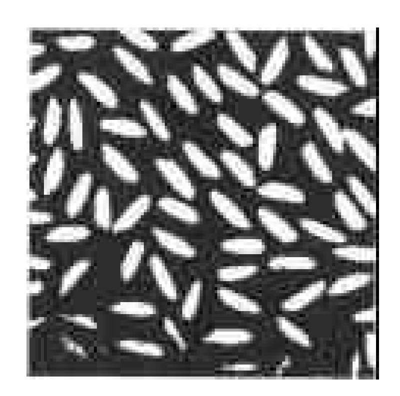



图 5.30 按照指定灰度范围选择图像区域结果

5.6.2 特定区域滤波

MATLAB 图像处理工具箱中提供的 roifilt2 函数用于对一个区域滤波，其语法格式为：

```
J=roifilt2(h,I,BW)
J=roifilt2(I,BW,fun)
J=roifilt2(I,BW,fun,P1,P2,…)
```

J=roifilt2(h,I,BW)使用滤波器 h 对图像 I 中用二值掩模 BW 选中的区域滤波。

J=roifilt2(I,BW,fun)和 J=roifilt2(I,BW,fun,P1,P2,…)对图像 I 中用二值掩模 BW 选中的区域作函数运算 fun，其中 fun 是描述函数运算的字符串，参数为 P1、P2……。返回图像 J 在选中区域的像素为图像 I 经 fun 运算的结果，其余部分的像素值为 I 的原始值。

下面的例子是对指定区域进行锐化滤波，结果如图 5.31 所示。

```
I=imread('eight.tif')
C=[121 272 300 20 21 194]
R=[51 31 45 121 21 75]
BW=roipoly(I,c,r)       %指定滤波区域为 c 和 r 确定的多边形。
H=fspecial('unsharp')    %指定滤波算子为'unsharp'。
```

```
J=roifilt2(h,I,BW)
subplot(1,2,1),imshow(I)
subplot(1,2,2),imshow(J)
```

从图像中可以看出，右上角的硬币发生变化，而其他硬币保持不变，由此可知，roifilt2 函数只对指定的区域进行滤波。

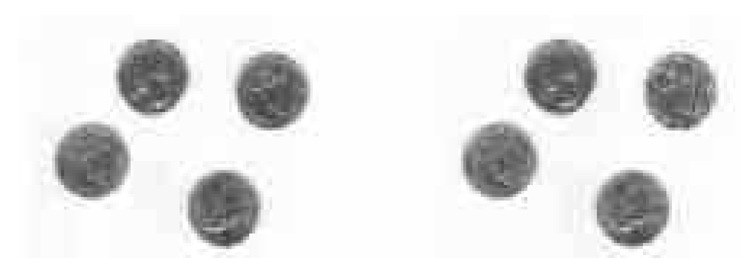


图 5.31　对选定区域进行滤波的结果

5.6.3　特定区域填充

MATLAB 图像处理工具箱中提供的 roifill 函数用于对特定区域的填充，其语法格式为：

```
J=roifill(I,c,r)
J=roifill(I)
J=roifill(I,BW)
[J,BW]=roifill(…)
J=roifill(x,y,I,xi,yi)
[x,y,J,BW,xi,yi]= roifill(…)
```

其中 J=roifill(I,c,r)填充由向量 c、r 指定的多边形，c 和 r 分别为多边形各顶点的 X、Y 坐标。它是通过解边界的拉普拉斯方程，利用多边形边界的点的灰度平滑的插值得到多边形内部的点。通常可以利用对指定区域的填充来“擦”掉图像中的小块区域。

J=roifill(I)由用户交互选取填充的区域。选择多边形的角点后，按 Enter 键键表示结束，空格键或 Del 键表示取消一个选择。

J=roifill(I,BW)用掩模图像 BW 选择区域。

[J,BW]=roifill(…)在填充区域的同时还返回掩模图像 BW。

J=roifill(x,y,I,xi,yi)和[x,y,J,BW,xi,yi]= roifill(…)在指定的坐标系 X—Y 下填充由向量 xi，yi 指定的多边形区域。

下面的例子为填充指定的区域，结果如图 5.32 所示。

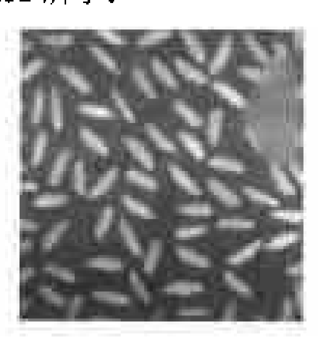



图 5.32　对指定区域进行填充的结果

```
I=imread('rice.tif')
C=[52 72 300 270 221 194]
R=[71 21 75 121 121 75 ]
J=roifill(I,c,r)
Subplot(1,2,1),imshow(I)
Subplot(1,z2,2),imshow(J)
```

第6章 图像变换

在图像处理技术中，图像的(正交)变换技术有着广泛的应用，是图像处理的重要工具。通过变换图像，改变图像的表示域及表示数据，可以给后继工作带来极大的方便。例如，傅立叶变换可使处理分析在频域中进行，使运算简单；而离散余弦变换(DCT)可使能量集中在少数数据上，从而实现数据压缩，便于图像传输和存储。

在 MATLAB 图像处理工具箱中，提供了几种常用的图像变换函数，它们是傅立叶变换(Fourier Transform)、离散余弦变换(Discrete Cosine Transform)和 Radon 函数变换(Radon Transfrom)。另外，随着小波分析方法在图像处理中的应用不断发展成熟，MATLAB 小波分析工具箱也提供了很多小波变换的函数，用于图像处理。

6.1 傅立叶变换

6.1.1 离散傅立叶变换

在图像处理的广泛应用领域中，傅立叶变换起着非常重要的作用，具体表现在包括图像分析、图像增强及图像压缩等方面。

假设 f(m,n)是一个离散空间中的二维函数，则该函数的二维傅立叶变换的定义如下：

$$f(\omega_1,\omega_2)=\sum_{-\infty}^{+\infty}\sum_{-\infty}^{+\infty}f(m,n)e^{-j\omega_1 m}e^{-j\omega_2 n}$$

其中 ω_1 和 ω_2 是频域变量，单位是弧度/采样单元。通常函数 $F(\omega_1,\omega_2)$称为函数 f(m,n)的频谱。$F(\omega_1,\omega_2)$是复变函数， ω_1 和 ω_2 的周期都是 2π。

二维傅立叶反变换的定义如下：

$$f(m,n)=\int_{\omega_1=-\pi}^{\pi}\int_{\omega_2=-\pi}^{\pi}F(\omega,\omega_2)e^{-j\omega_1 n}e^{-j\omega_2 n}d\omega_1\omega_2$$

这个式子表明，函数 f(m,n)可以用无数个不同频率的复指数信号的和表示，而在频率(ω_1,ω_2)处复指数信号的幅度和相位是 $F(\omega_1,\omega_2)$。

例如，函数 f(m,n)在一个矩形区域内函数值为 1，而在其他区域为 0，如图 6.1 所示。

为了简便起见，假设 f(m,n)为一个连续函数，则 f(m,n)的傅立叶变换的幅度值(即$|F(\omega_1,\omega_2)|$)显示为网格图，如图 6.2 所示。

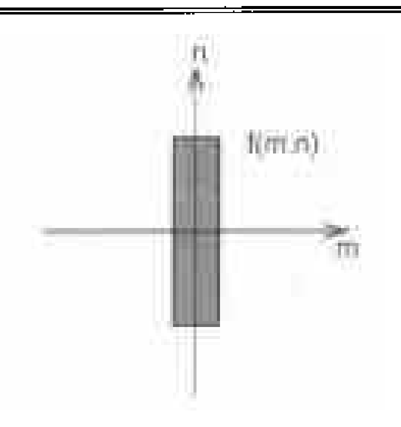


图 6.1 离散矩形函数

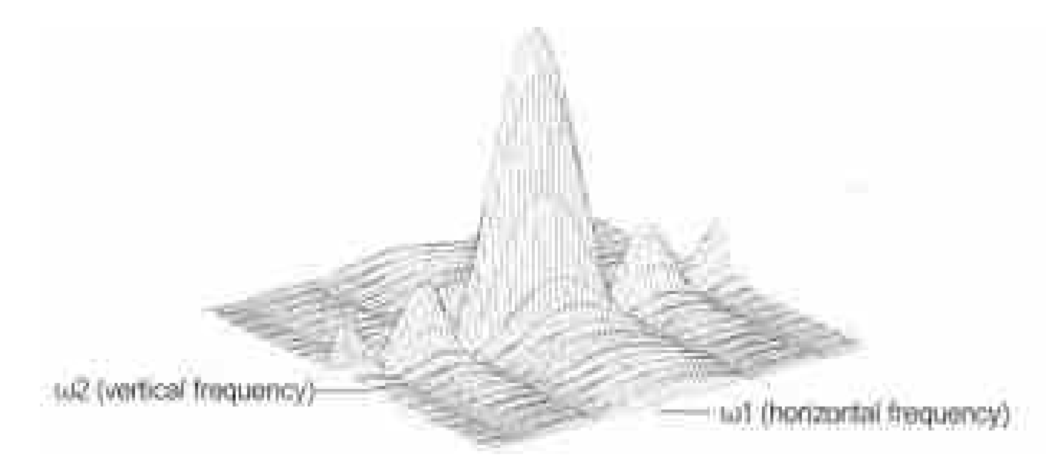


图 6.2 函数f(m,n)的傅立叶变换幅度的三维显示

将傅立叶变换的结果进行可视化的另一种方法是用图像的方式显示变换结果的对数幅值 $\log|F(\omega_1,\omega_2)|$，如图 6.3 所示。

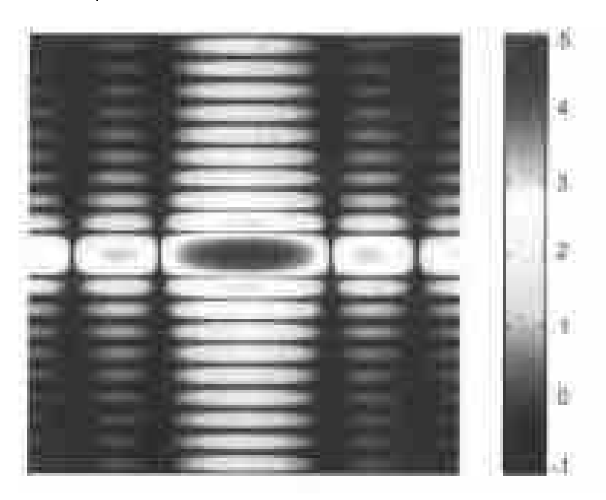


图 6.3 傅立叶变换幅度对数的二维显示

几种简单函数的傅立叶变换的频谱可以直观地表示为图 6.4 所示的样子。

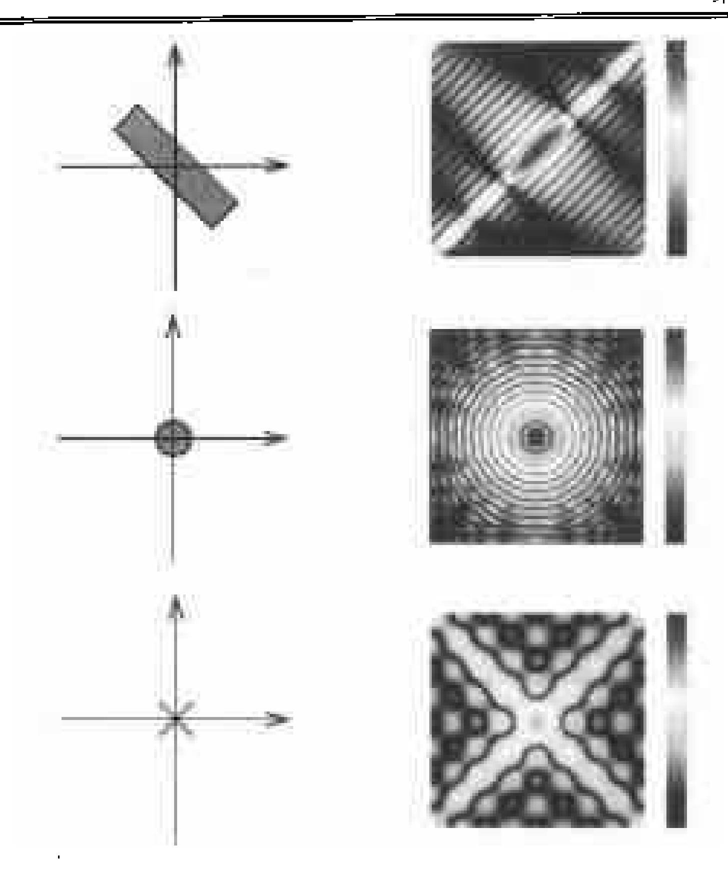


图 6.4 几种简单函数的傅立叶变换

利用计算机进行傅立叶变换的通常形式为离散傅立叶变换，采用这种形式的傅立叶变换有以下两个原因：

- 离散傅立叶变换的输入输出都是离散值，适用于计算机的运算操作。
- 采用离散傅立叶变换，可以应用快速傅立叶变换来实现，提高运算速度。

离散傅立叶变换的定义如下：

$$F(p,q)=\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}f(m,n)e^{-j(2\pi/M)pm}e^{-j(2\pi/N)pn} \qquad p=0,1,\ldots M-1 \qquad q=0,1,\ldots N$$

离散傅立叶反变换的定义如下：

$$f(m,n)=\sum_{p=0}^{M-1}\sum_{q=0}^{N-1}F(p,q)e^{j(2\pi/M)pm}e^{j(2\pi/N)qn} \qquad m=0,1,\ldots M-1 \qquad n=0,1,\ldots N-1$$

F(p,q)称为 f(m,n)的离散傅立叶变换系数。

从函数取值角度考虑，$F(p,q)$和 $F(\omega_1,\omega_2)$的关系为：

$$F(p,q)=F(\omega_1,\omega_2)\big|_{\substack{\omega_1=2\pi p/M\\ \omega_2=2\pi q/N}} \qquad \begin{matrix} p=0,1\ldots M-1\\ q=0,1\ldots N-1\end{matrix}$$

也就是说，F(p,q)是函数傅立叶变换在圆周上等间距的采样，采样间隔等于 2π/M，M 为离散傅立叶变换的点数。

下面比较一下用不同的点数实现离散傅立叶变换的效果：将前面提到的矩形函数分别进行 30×30 点 DFT 和 256×256 点 DFT，并将变换的直流分量移至原点，则变换结果如图

6.5 所示。

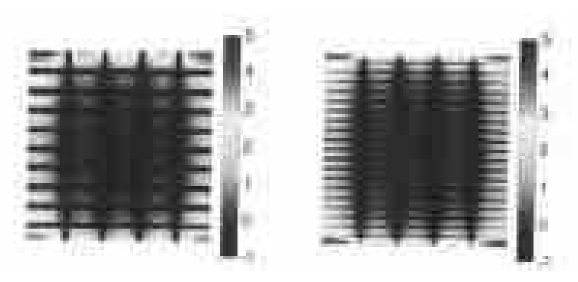


图 6.5 30×30点DFT和256×256点DFT结果

可以看出 256×256 点 DFT 比 30×30 点 DFT 看起来更平滑，这说明离散傅立叶变换的点数越多，其在频域中采用间的隔越细，计算的结果越接近离散函数傅立叶变换的真实形状。

二维离散傅立叶变换有一些重要性质，如表 6.1 所示。

表6.1 傅立叶变换的性质及表达式

性　质	表 达 式
周期性	$F(u,v)=F(u+mM,v+nN)$
线性	$F[a_1f_1(x,y)+a_2f_2(x,y)]=a_1F[f_1(x,y)]+a_2F[f_2(x,y)]$
可分离性	$F(u,v)=F_x[F_y[f(x,y)]]=F_y[F_x[f(x,y)]]$
比例性质	$f(ax+by)=\frac{1}{\|ab\|}F\left(\frac{u}{a},\frac{v}{b}\right)$
位移性质	$f(x-x_0,y-y_0)\leftrightarrow F(u,v)e^{-2\pi(ux_0+vy_0)/N}$ $f(x,y)e^{2\pi(u_0x+v_0y)/N}\leftrightarrow F(u-u_0,v-v_0)$
对称性	$f(x,y)=f(-x,-y)\Rightarrow F(u,v)=F(-u,-v)$
共轭对称性	$f^*(x,y)\leftrightarrow F^*(-u,-v)$
差分	$f(x,y)-f(x-1,y)\leftrightarrow(1-e^{-j2\pi u/N})F(u,v)$
积分	$f(x,y)+f(x-1,y)\leftrightarrow(1+e^{-j2\pi u/N})F(u,v)$
卷积	$f(x,y)*g(x,y)\leftrightarrow F(u,v)G(u,v)$ $f(u,v)\times g(x,y)\leftrightarrow F(u,v)G(u,v)$
能量	$\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\|f(x,y)\|^2=\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}\|F(u,v)\|^2$

6.1.2 MATLAB 提供的快速傅立叶变换函数

在 MATLAB 中提供了函数 fft，函数 fft2 和函数 fftn 分别用于进行一维 DFT、二维 DFT 和 n 维 DFT，函数 ifft、函数 ifft2 和函数 ifftn 分别用于进行一维 DFT、二维 DFT 和 n 维 DFT 的反傅立叶变换。

1. fft2

fft2 函数用于计算二维快速傅立叶变换，其语法格式为：

```
B=fft2(I)
B=fft2(I,m,n)
```

其中 B=fft2(I)返回图像 I 的二维 fft 变换矩阵，输入图像 I 和输出图像 B 大小相同。

B=fft2(I,m,n)通过对图像 I 剪切或补零，按用户指定的点数计算 fft，返回矩阵 B 的大小为 m×n。很多 MATLABL 图像显示函数无法显示复数图像，为了观察图像傅立叶变换后的结果，应对变换后的结果求模，方法是对变换结果调用 abs()函数。

例如，计算图像的二维傅立叶变换，并显示其幅值的结果，如图 6.6 所示，其命令格式如下：

```
load imdemos saturn2
imshow(saturn2)
B=fftshift(fft2(saturn2))
figure, imshow(log(abs(b)),[]);
colormap(jet(64)),
colorbar
```




图 6.6 原始图像及其傅立叶谱

2. fftn

fftn 函数用于计算 n 维傅立叶变换，其语法格式为：

```
B=fftn(I)
B=fftn(I,siz)
```

B=fftn(I)计算图像的 n 维傅立叶变换，输出图像 B 与 I 大小相同。

B=fftn(I,siz)函数通过对图像 I 剪切或补零，按 siz 指定的点数计算给定矩阵的 n 维 fft，返回矩阵 B 的大小也是 siz。

3. fftshift

MATLAB 提供的 fftshift 函数用于将变换后的图像频谱中心从矩阵的原点移到矩阵的中心，其语法格式为：

```
B=fftshift(I)
```

fftshift 可以用于调整 fft、fft2 和 fftn 的输出结果。对于向量，fftshift(I)将 I 的左右两半交换位置，对于矩阵 I，fftshift(I)将 I 的一、三象限和二、四象限进行互换，对于高维矢量，fftshift(I)将矩阵各维的两半进行互换。

4. ifft2

ifft2 函数用于计算图像的二维傅立叶反变换，其语法格式为：

```
B=ifft2(I)
B=ifft2(I,m,n)
```

B=ifft2(I)返回图像 I 的二维傅立叶反变换变换矩阵，输入图像 I 和输出图像 B 大小相同。B=ifft2(I,m,n)通过对图像 I 剪切或补零，按用户指定的点数计算二维傅立叶反变换，返回矩阵 B 的大小为 m×n。通常输出矩阵 B 为复数矩阵，如果要求模，需用 abs()函数。

函数 iffi2 的语法格式含义与 fft 函数的语法格式相同，可参考本节对 fft2 函数的说明。

5. ifftn

ifftn 函数用于计算 n 维傅立叶逆变换，其语法格式为：

```
B=ifftn(I)
B=ifftn(I,siz)
```

B=ifftn(I)计算图像的 n 维傅立叶反变换，输出图像 B 与 I 大小相同。B=ifftn(I,siz)函数通过对图像 I 进行剪切或补零，按用户指定的点数计算傅立叶反变换，返回矩阵 B 的大小为 siz。

6.1.3 快速傅立叶变换的应用

本节介绍傅立叶变换在图像处理中的几个应用。

1. 滤波器频率响应

由线性系统理论可知，滤波器冲激响应的傅立叶变换就是该滤波器的频率响应。MATLAB 工具箱中提供的 freqz2 函数可以同时计算和显示滤波器的频率响应。拉普拉斯高斯滤波器的频率响应如图 6.7 所示，它的频率响应体现了高斯函数的低通特性。

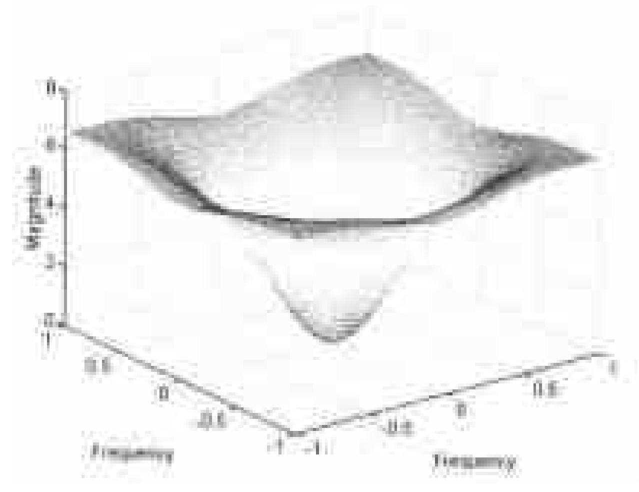


图 6.7 拉普拉斯高斯滤波器的频率响应

```
h=fspecial('log')
freqz2(h)
```

2.　快速卷积

傅立叶变换的另一个重要特性是能够实现快速卷积。由线性系统理论可知，两个函数卷积的傅立叶变换等于两个函数的傅立叶变换的乘积。该特性与快速傅立叶变换一起，可以快速计算函数的卷积。

假设 A 为 M×N 矩阵，B 为 P×Q 矩阵，则快速计算卷积的方法如下：

- 对 A 和 B 补 0 ，使其大小都为(M+P－1)×(N+Q－1)。
- 利用 fft2 的矩阵 A 和 B 进行二维 FFT 变换。
- 将两个 FFT 结果相乘，利用 ifft2 对得到的乘积进行傅立叶反变换。

例如 A=magic(3)

```
A =
     8     1     6
     3     5     7
     4     9     2
B=ones(3)
B =
     1     1     1
     1     1     1
     1     1     1
A(8,8)=0          %为了进行快速卷积，将矩阵 A 补零。
B(8,8)=0          %为了进行快速卷积，将矩阵 B 补零。
C=ifft2(fft2(A).*fft2(B))
C=C(1:5,1:5)
C=real(C)
%得到的 A 与 B 的卷积结果为
C =
     8.0000     9.0000    15.0000     7.0000     6.0000
    11.0000    17.0000    30.0000    19.0000    13.0000
    15.0000    30.0000    45.0000    30.0000    15.0000
     7.0000    21.0000    30.0000    23.0000     9.0000
     4.0000    13.0000    15.0000    11.0000     2.0000
```

同时利用 MATLAB 提供的卷积函数 conv2 进行验证，可知结果正确。

```
C=conv2(A,B)
C=C(1:5,1:5)
C =
     8     9    15     7     6
    11    17    30    19    13
    15    30    45    30    15
     7    21    30    23     9
     4    13    15    11     2
```

3.　图像特征识别

傅立叶变换可以用于与卷积密切相关的相关运算(correlation)。在数字图像处理中相关

的运算通常用于匹配模板，可以用于对某些模板对应的特征进行定位。例如，假如我们希望在图像 text.tif 中定位字母“a”，如图 6.8 所示，可以采用下面的方法定位。

将包含字母“a”的图像与 text.tif 图像进行相关运算，也就是首先将字母 a 和图像 text.tif 进行傅立叶变换，然后利用快速卷积的方法，计算字母 a 和图像 text.tif 的卷积(其结果如图 6.9 所示)，提取卷积运算的峰值，如图 6.10 所示的白色亮点，即得到在图像 text.tif 中对字母“a”定位的结果。



图 6.8 图像text.tif和字母“a”

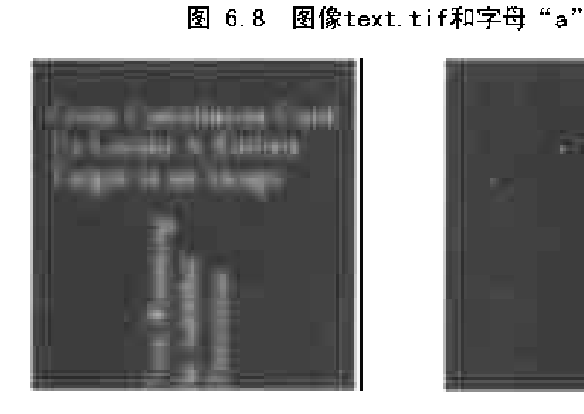




图 6.9 快速卷积结果　　图 6.10 对字母“a”定位的结果

具体实现的方法为，首先将字母‘a’切割出来，存为文件 a.tif，然后执行下面的程序代码：

```
a=imread('.tif');  %读入字母 a。
[ma,na]=size(a);   %求出 a 的大小。
I=imread('text.tif');  %读入图像'text.tif'。
[mi,ni]=size(I);  %求出图像的大小。
afft=fft2(a) ;
Ifft=fft2(I);
M=ma+mi-1;
N=na+ni-1;  %求出卷积之后图像的大小。
afft(M,N)=0;  %对字符 a 的 fft 变换补零，以利于快速卷积。
Ifft(M,N)=0;  %对 text 图像的 fft 变换补零，以利于快速卷积。
filtered=ifft2(afft.*Ifft);  %利用 fft 和卷积的关系快速计算矩阵卷积。
filtered=filtered(1:mi,1:ni);  %只保留原图像大小。
filtered=filtered/max(max(filtered,[],1));  %对卷积之后的图像归一化。
result=filtered>0.9;  %设置门限求出匹配点。
imshow(result);
```

6.2 离散余弦变换

在 MATLAB 图像处理工具箱中，dct2 函数用于计算图像的二维离散余弦变换(Discrete Cosine Transform)，简称 DCT。大多数情况下，DCT 用于压缩图像，JPEG 图像格式就采用了 DCT 算法。

6.2.1 离散余弦变换的定义

二维离散余弦变换的定义为，假设矩阵 A 的大小为 M×N，

$$B_{p,q}=\alpha_p\alpha_q\sum\sum A_{mn}\cos\frac{\pi(2m+1)p}{2M}\cos\frac{\pi(2n+1)p}{2N}\qquad \begin{matrix}p=0,1\ldots M-1\\ q=0,1\ldots N-1\end{matrix}$$

$$\alpha_p=\begin{cases}1/\sqrt{M},p=0\\ \sqrt{2/M},1\le p\le M-1\end{cases}\qquad \alpha_q=\begin{cases}1/\sqrt{N},q=0\\ \sqrt{2/N},1\le q\le N-1\end{cases}$$

其中 $B_{p,q}$ 称为矩阵 A 的 DCT 系数。在 MATLAB 中，矩阵的下标从 1 开始而不是从 0 开始，所以 A(1,1)和 B(1,1)分别代表上面的 A_{00} 和 $B_{0,0}$。

DCT 是一种可逆变换，它的逆变换定义为：

$$A_{mn}=\sum\sum\alpha_p\alpha_q B_{p,q}\cos\frac{\pi(2m+1)p}{2M}\cos\frac{\pi(2n+1)q}{2N}\qquad \begin{matrix}m=0,1,\ldots M-1\\ n=0,1,\ldots N-1\end{matrix}$$

$$\alpha_p=\begin{cases}1/\sqrt{M},p=0\\ \sqrt{2/M},1\le p\le M-1\end{cases}\qquad \alpha_q=\begin{cases}1/\sqrt{N},q=0\\ \sqrt{2/N},1\le q\le N-1\end{cases}$$

上式的含义是任何 M×N 的矩阵 A 都可以表示为一系列函数的和：

$$\alpha_p\alpha_q B_{p,q}\cos\frac{\pi(2m+1)p}{2M}\cos\frac{\pi(2n+1)q}{2N}\qquad \begin{matrix}p=0,1,\ldots M-1\\ q=0,1,\ldots N-1\end{matrix}$$

这些函数称为 DCT 变换的基函数。

MATLAB 图像处理工具箱提供的 DCT 变换函数有以下几种。

1. dct2

dct2 函数实现图像的二维离散余弦变换，其语法格式为：

```
B=dct2(A)
B=dct2(A,m,n)
B=dct2(A,[m n])
```

B=dct2(A)返回图像 A 的二维离散余弦变换值，它的大小与 A 相同，且各元素为离散余弦变换的系数 B(k1,k2)。

B=dct2(A,m,n)或者 B=dct2(A,[m n])在对图像 A 进行二维离散余弦变换之前，先将图像

A 补零至 m×n。如果 m 和 n 比图像 A 小，则进行变换之前，将图像 A 剪切。

例如下例将如图 6.11 所示的一幅秋天的图像进行 DCT 变换，可以注意到图像能量的绝大部分位于变换矩阵的左上角，如图 6.12 所示。

```
RGB=imread('autumn.tif');
I=rgb2gray(RGB);         %将彩色图像转化为灰度图像，以便进行 DCT 变换。
imshow(I);
J=dct2(I);
imshow(log(abs(J)),[]),
colormap(jet(64)),
colorbar;
```



图 6.11 原始图像

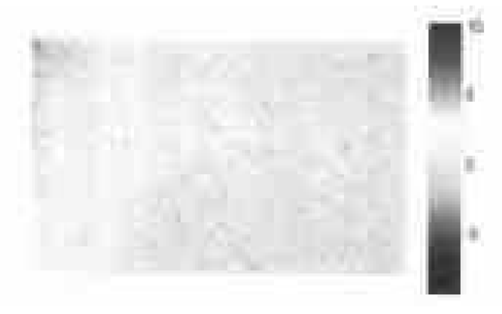


图 6.12 图像DCT变换的二维显示

2. idct2

idct 函数可以实现图像的二维离散余弦反变换，其语法格式为：

```
B=idct2(A)
B=idct2(A,m,n)
B=idct2(A,[m n])
```

B=idct2(A)计算矩阵 A 的二维离散余弦反变换，返回图像 B 的大小与 A 相同。

B=idct2(A,m,n)或者 B=idct2(A,[m n])在对矩阵 A 进行二维离散余弦反变换之前，先将矩阵 A 补零至 m×n。如果 m 和 n 比矩阵 A 小，则进行变换之前，先对矩阵 A 进行剪切操作，返回图像大小为 m×n。

下面将 DCT 变换值小于 10 的系数设为 0，然后利用 idct2 函数重构图像，结果如图 6.13 所示。

```
J(abs(J)<10)=0;          %将 DCT 变换值小于 10 的元素设为 0。
K=idct2(J)/255;          %对逆 DCT 变换值归一化。
figure,imshow(K)
```



图 6.13 对压缩图像重构结果

从重构图像可以看出，离散余弦变换用于图像压缩，压缩图像的质量比较令人满意。

6.2.2 离散余弦变换和图像压缩

在 JPEG 图像压缩算法中，图像被分成 8×8 或者 16×16 的图像块，然后对每个图像块进行 DCT 变换。DCT 变换被量化、编码及传输。在接收端，量化的 DCT 系数被解码，并用来计算每个图像块的逆 DCT 变换，最后把各图像块拼接起来构成一幅图像。对一幅典型的图像而言，许多 DCT 变换的系数近似为 0，把它们去掉并不会明显影响重构图像的质量。

下例展示了如何把图 6.14 所示的输入图像划分成 8×8 的图像块，计算它们的 DCT 系数，并且只保留 64 个 DCT 系数中的 10 个，然后对每个图像块利用这 10 个系数进行逆 DCT 变换来重构图像，结果如图 6.15 所示。

```
I=imread('cameraman.tif');
I=im2double(I);
T=dctmtx(8);            %产生二维 DCT 变换矩阵。
B=blkproc(I,[8 8],'P1*x*P2',T,T');      %计算二维 DCT
mask=[1 1 1 1 0 0 0 0
    1 1 1 0 0 0 0 0
    1 1 0 0 0 0 0 0
    1 0 0 0 0 0 0 0
    0 0 0 0 0 0 0 0
    0 0 0 0 0 0 0 0
    0 0 0 0 0 0 0 0
    0 0 0 0 0 0 0 0];
%二值掩模，用来压缩 DCT 的系数
B2=blkproc(B,[8 8],'P1.*x',mask);
%只保留 DCT 变换的 10 个系数。
I2=blkproc(B2,[8 8],'P1*x*P2',T',T);
%逆 DCT 变换，用来重构图像。
imshow(I);
figure,imshow(I2)
```



图 6.14 原始图像



图 6.15 压缩重构图像

程序中利用了 MATLAB 提供的函数 dctmtx，它用于计算二维离散 DCT 矩阵，其语法

格式为：

```
D=dctmtx(n)
```

D=dctmtx(n)返回 n×n 的 DCT 变换矩阵，如果矩阵 A 的大小为 n×n，D*A 是 A 矩阵每一列的 DCT 变换值，D'*A 是 A 每一列的逆 DCT 变换值。

如果矩阵 A 为 n×n 的方阵，则 A 的 DCT 变换可以用来 D*A*D'计算。这有时候比利用 dct2 计算二维离散 DCT 要快，特别是对于 A 很大的情况，因为 D 只需要计算一次即可。

6.3 Radon 变换

6.3.1 Radon 变换的定义

图像处理工具箱的 radon 函数用来计算指定方向上图像矩阵的投影，二元函数 f(x，y) 的投影是在某一方向上的线积分，例如 f(x，y)在垂直方向上的线积分是 f(x，y)在 x 方向上的投影，在水平方向上的积分是在 y 方向上的投影。图 6.16 是一个简单的二维函数在水平垂直方向的投影。

推而广之，投影可沿任意角度 θ 进行，通常 f(x,y)的 radon 变换是 f(x,y)平行于 y' 轴的线积分，格式如下：

$$R_\theta(x') = \int_{-\infty}^{\infty} f(x'\cos\theta - y'\sin\theta, x'\sin\theta + y'\cos\theta)dy'$$

图 6.17 表示任意角度 radon 变换的几何关系。

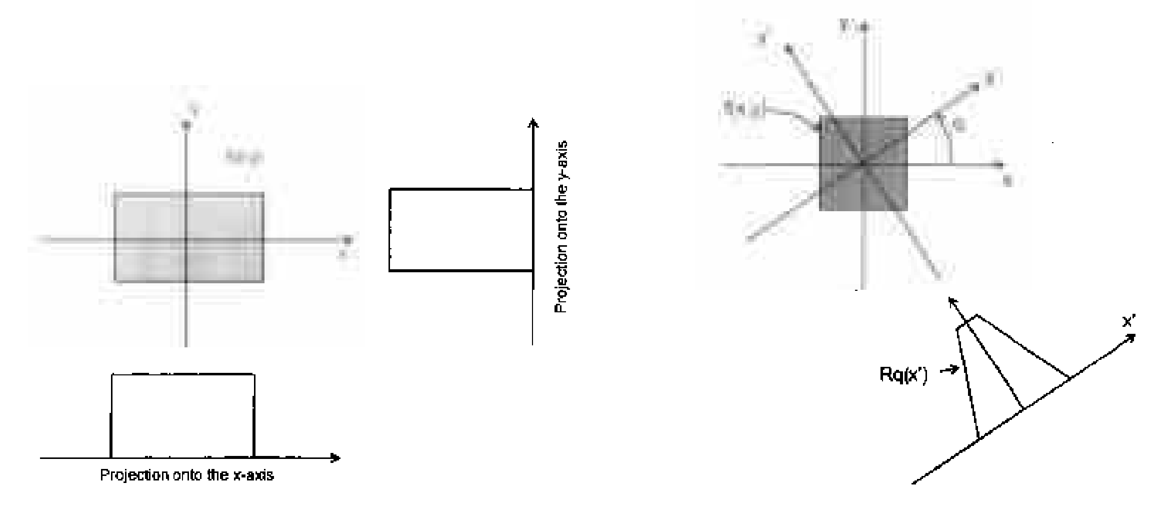


图 6.16　矩形函数在水平垂直方的投影　　　图 6.17　任意角度radon变换的几何关系

下面的命令计算图像 I 在 theta 矢量指定的方向上的 radon 变换。

```
[R,xp]=radon(I,theta)
```

R 的各行返回 theta 中各方向上的 radon 变换值，xp 矢量表示沿 x'轴相应的坐标值。图像 I 的中心在 floor((size(I)+1)/2)，在 x'轴上对应 x'＝0。

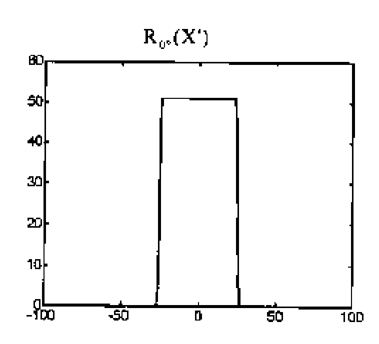
计算并画出如图 6.18 所示的含有正方形图像在 0° 和 45° 方向上的 radon 变换命令如下：

```
I=zeros(100,100);
I(25:75,25:75)=0;      %产生一个正方形的黑框。
imshow(I)
[R,xp]=radon(I,[0 45]);   %计算黑框的 radon 变换。
figure, plot(xp,R(:, 1)), title('R_{0^o}(x\prime)');
%显示黑框在 0° 的 radon 变换。
figure, plot(xp, R(:,2)), title('R_{45^o}(x\prime)');
%显示黑框在 45° 的 radon 变换。
```



图 6.18　方框图像

方框图像在 0° 和 45° 的 radon 变换的图如图 6.19 所示。



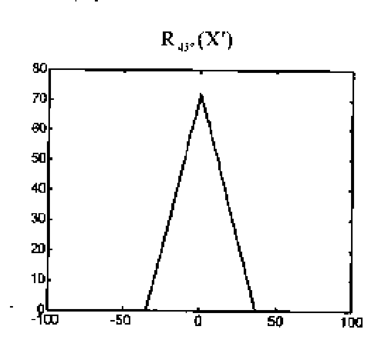


图 6.19　方框图像在0° 和45° radon变换

注意对任意投影方向 xp 相同。

对于有很多角度的 radon 变换通常可以表示为图像。计算方形图像从 0° 到 180° 每隔 1° 计算一次 radon 变换的命令如下，其结果如图 6.20 所示。

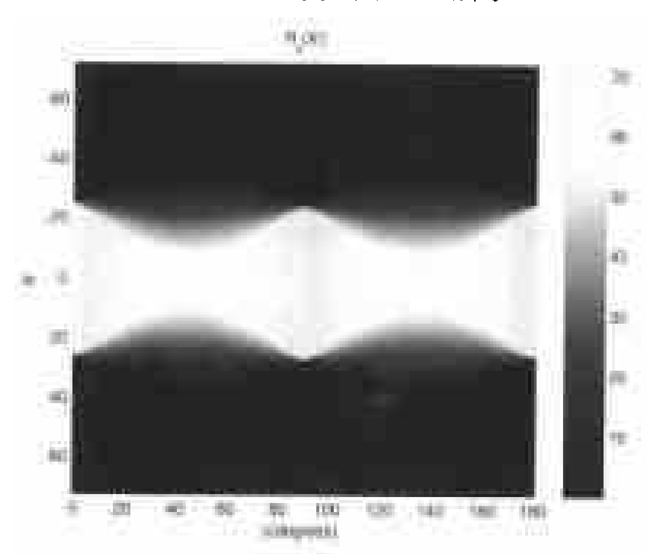


图 6.20　方框图像在0° ~180° radon变换

```
I=zeros(100,100);
I(25:75,25:75)=1;      %产生一个正方形的黑框。
theta=0:180;      %产生 0° ~180° 的角度向量。
[R,xp]=radon(I,theta);
imagesc(theta,xp,R);
```

```
title('R_{\theta}(X\prime)');
xlabel('\theta(degrees)');
ylabel('X\prime')
set(gca,'XTick',0:20:180);    %设置坐标轴属性。
colormap(hot);
colorbar
```

6.3.2 利用 radon 变换检测直线

radon 变换与计算机视觉中的 hough 变换很相似，我们可以利用 radon 变换来实现 hough 变换，方法是：

- 用 edge 函数计算如图 6.21 所示图像的边缘的二值图像，结果如图 6.22 所示，其命令如下：

```
I= imread('ic.tif');
BW=edge(I);       %求出图像 I 的边缘点。
imshow(I)
figure, imshow(BW)
```

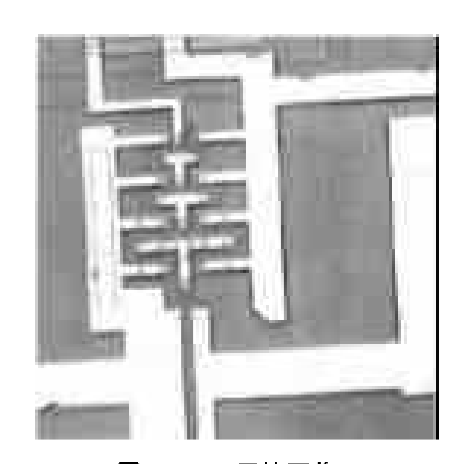


图 6.21 原始图像



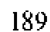
图 6.22 边缘图像

- 计算边缘图像的 radon 变换，结果如图 6.23 所示。

```
theta=0:179;
[R, xp]=radon(BW,theta);     %计算出二值边缘点的 radon 变换。
figure, images(theta,xp,R),  colormap(hot)
xlabel('x'\theta(degrees)'),  ylabel('x\prime');
title('R{\theta}(x\prime)');
colorbar
```

- 计算出 radon 变换矩阵中的峰值，这些峰值对应于原始图像中的直线。这个例子中，R 的最强值出现在 θ=94°，x'=−101 处；垂直于 θ=94°，并位于 x'=−101 的直线显示在下图，用红色覆盖在原始图像上，radon 变换的几何关系如图 6.24 所示。

注意到平行于红线的直线也在 θ=94° 处有峰值，垂直于红线的直线峰值在 θ=−4° 处。



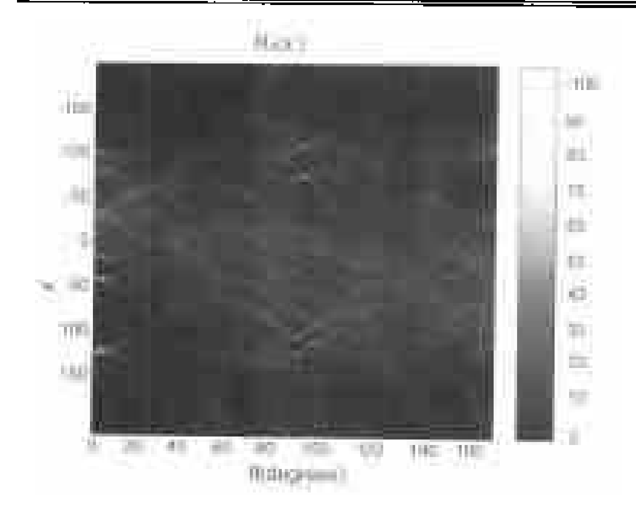


图 6.23　边缘图像的radon变换

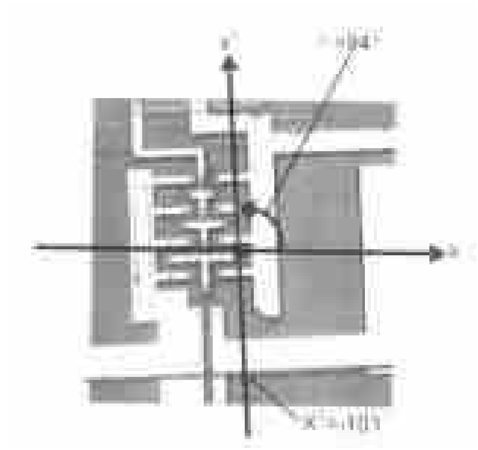


图 6.24　原始图像中的直线

6.3.3　逆 Radon 变换及应用

iradon 函数可以实现逆 radon 变换，并经常用于投影成像中。这个变换能把 radon 变换反变换回来，因此可以从投影数据重建原始图像。

6.3.2 节 radon 变换中指出，给定图像 I 和一系列角度 theta，radon 函数可以用来计算图像的 radon 变换，表达式为 R=radon(I,theta)。在 MATLAB 中也可以利用 iradon 函数重建图像，方法如下：

```
R=iradon(R,theta)
```

在这里，图像的 radon 变换(即投影)可以从图像中计算出来。而在大多数应用中，没有所谓的原始图像来计算投影。例如 X 射线吸收重建，投影是通过测量 X 射线辐射在不同角度通过物理切片时的衰减得到的。原始图像可以认为是通过切面的截面。这里图像的密度代表切片的密度。投影通过特殊的硬件设备获得，而切片内部图像通过 iradon 重建。这可以用来对活的生物体或者不透明物体实现无损成像。

iradon 函数通过平行波束的投影来重建图像。在平行波束的几何关系中，每个投影通过把特定角度穿过图像的一系列线积分组合得到。

图 6.25 显示了 X 射线投影重建中，平行波束的几何关系。注意到有相同数目的辐射器和接收器，辐射的衰减代表物体的密度、质量等的积分，这相当于 radon 变换中的线积分。

本图照射波束的关系是平行的，与 radon 变换的几何关系相同。f(x, y)代表图像亮度，$R_\theta(x')$代表 θ 方向的投影。

另外一种几何关系是扇形波束，即只有一个辐射器而有多个接收器，扇形波束投影可以转换成平行波束投影，利用逆 radon 变换来重建图像。

iradon 变换利用滤波后向投影算法来计算逆 radon 变换。这种算法利用 R 各列中的投影来构造图像 I 的近似值，若要获得更准确的图像，可以使用更多的投影值，投影数越多，重建的图像越接近原始图像。theta 矢量必须是固定增量的均匀矩阵，即每次角度的增加值 $\Delta\theta$ 为常数，若 $\Delta\theta$ 已知，可作为参数取代 theta 值，传入 iradon 函数。例如：

```
IR=iradon(R,delta)
```

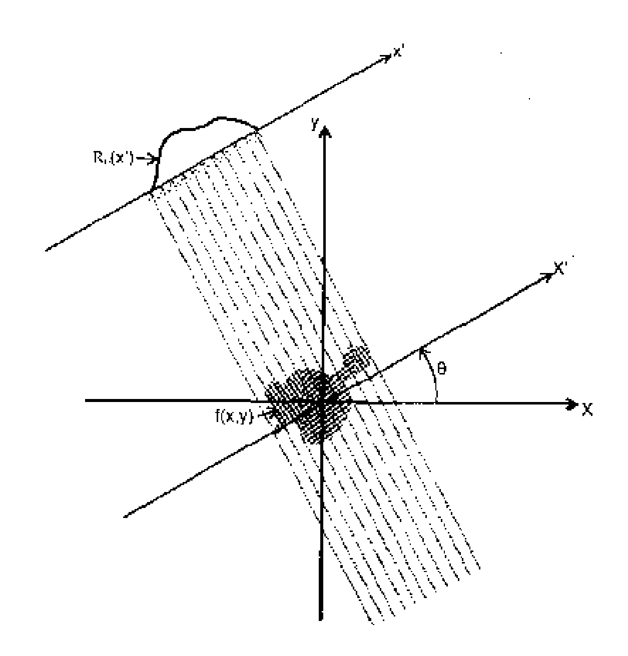


图 6.25 逆radon变换的几何关系

滤波 radon 变换是先对投影 R 滤波，再用滤波后的投影值重构图像。有些情况下，投影值含有噪声，为了消除高频噪声，可以通过加窗去除噪声。iradon 函数中可以采用多种窗函数，下面的例子采用了 hamming 窗来滤波。

```
IR=iradon(R,theta,'hamming')
```

iradon 也允许指定归一化频率 D，高于 D 的滤波器响应为 0，整个滤波器压缩在[0, D]范围内。这在投影不含高频信息而存在高频噪声的条件下很有用。这时，噪声完全被抑制，而不会影响图像重建。下例调用 iradon，设定归一化频率为 0.85。

```
IR=iradon(R,theta,0.85)
```

接下来介绍如何利用 radon 函数和 iradon 函数构造一个简单图像的投影并重建图像。

测试图像是 Shepp—Logan 的大脑图，利用图像处理工具箱的 phantom 函数可以产生数据。Shepp—Logan 的大脑图反映了真实世界中人类大脑的很多性质。图像中外部的椭圆形是头骨，内部的椭圆是大脑的内部特征或者是肿瘤。

首先产生如图 6.26 所示的 256 个灰度等级的大脑图，命令如下：

```
P=phantom(256);
imshow(P)
```

然后计算 3 个不同部分的大脑图的 radon 变换，R1 有 18 个角度，R2 有 36 个角度，R3 有 90 个角度：

```
theta1=0:10:170;[R1,xp]=radon(P,theta1);
%计算 R1 的 radon 变换。
theta1=0:5:175;[R2,xp]=radon(P,theta2);
%计算 R2 的 radon 变换。
theta1=0:2:178;[R3,xp]=radon(P,theta3);
```

```
%计算 R3 的 radon 变换。
```

现在观察 Shepp—Logan 的大脑图 90 个角度的 radon 变换，如图 6.27 所示。



图 6.26 Shepp—Logan的大脑图

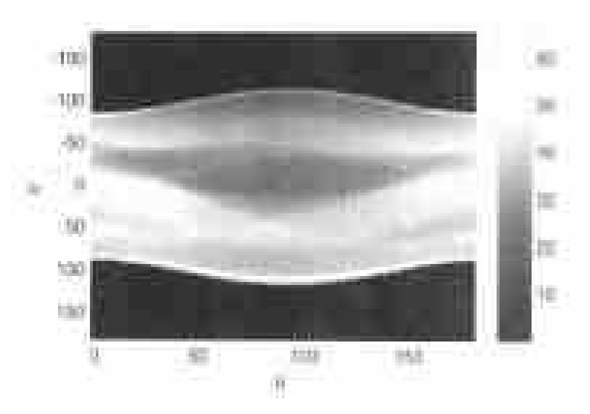


图 6.27 Shepp—Logan的大脑图90个角度的radon变换

观察上图可以看出输入图像的一些特性。图像 radon 变换的第一列为 0°，对应图像在垂直方向的投影，中间为 90°，对应图像在水平方向的投影。90° 的投影比 0° 的投影范围要大，这是因为在 90° 的水平方向，大脑的椭圆有最大的半轴。

下面用不同部分的逆 radon 变换来重构图像。如图 6.28 所示。可以发现用 R1 重构图像效果最差，因为它用来重构图像的投影最少。用 R2 重构图像效果较好，用 R3 重构图像质量最好。从图中可以看出，由于用 R1 重构图像投影太少，重构图像有很多的虚假点。为了避免这种情况，可以增加重构图像的投影角度的数目。



用R1重构图像结果



用R2重构图像结果



用R2重构图像结果

图 6.28 重构图像

6.4 离散小波变换

6.4.1 小波变换的定义及性质

设 $x(t)$是平方可积函数，即 $x(t)=L^2(R)$，$\psi(t)$是被称为基本小波或母小波(mother wavelet)的函数，定义如下：

$$WT_X(a,\tau)=\frac{1}{\sqrt{a}}\int x(t)\psi^*\left(\frac{t-\tau}{a}\right)dt=\langle x(t),\psi_{a\tau}(t)\rangle$$

为 $x(t)$的小波变换，式中 $a>0$ 是尺度因子，τ 代表位移。符号$<x,y>$代表内积，它的定义为：

$$\langle x(t),y(t)\rangle=\int x(t)y^*(t)dt\ ,$$

上标*代表共轭，$\psi_{a\tau}(t)=\frac{1}{\sqrt{a}}\psi\left(\frac{t-\tau}{a}\right)$是基本小波的位移和尺度伸缩，其中$t,a$与$\tau$都是连续变量，因此称上述定义是连续小波变换(continuous wavelet transform，简记为 CWT)。

连续小波变换有明确的物理意义，尺度因子 a 愈大，则$\psi\left(\frac{t}{a}\right)$愈宽，该函数的时间分辨率愈低。$\psi_{a\tau}(t)$前增加因子$1/\sqrt{a}$是为了使不同的$a$下的$\psi_{a\tau}(t)$能量相同。而 $WT_x(a,t)$在频域可以表示为：$WT(a,\tau)=\frac{\sqrt{a}}{2\pi}\int X(\omega)\,\psi^*(\omega)e^{j\omega\tau}d\omega$。$\psi(\omega)$是幅频特性比较集中的带通函数，小波变换具有表征分析信号$X(\omega)$频域上局部性质的能力。采用不同的 a 值作处理时，各$\psi(\omega)$的中心频率和带宽都不同，但品质因数(中心频率/带宽)却不变。

多分辨分析是小波分析的一个基石，它是在 $L^2(R)$函数空间内，将函数描述为一系列近似函数的极限，每一个近似函数都是函数 f 的平滑逼近，而且具有越来越精细的近似(Approximation)函数。这些近似都是在不同分辨水平(尺度)上得到的，因此称为多分辨分析或者多尺度分析。多分辨分析提供了寻求小波滤波器的基本思路：为了寻求 $L^2(R)$的一个基底，先从其某个子空间出发，构造它的基底，然后通过简单变换将之扩充至 $L^2(R)$中。我们首先给出多分辨分析的基本概念。

多分辨分析的定义为：

$L^2(R)$ 一系列嵌套子空间函数$V_j, j\in Z$，$\cdots\subset V_{-2}\subset V_{-1}\subset V_0\subset V_1\subset V_2\subset\cdots$，具有以下特点：

单调性：$V_j\subset V_{j+1}$，对任意的$j\in Z$。

- 逼近性：$\bigcup_{j\in Z}V_j$的闭包为$L^2(R),\bigcap_{j\in Z}V_j=\{0\}$。
- 伸缩性：$f(x)\in V_j\Leftrightarrow f(2x)\in V_{j+1}$。
- 平移性：$f(x)\in V_j\Leftrightarrow f\left(x+\frac{1}{2^j}\right)\in V_j$。
- Reisz 基：存在$g\in V_0$，使得$\{g(x-k)\mid k\in Z\}$构成V_0的 Reisz 基。即对任何$u\in V_0$，存在唯一序列$\{a_k\}\in l^2$，使得$u(x)=\sum_{k\in Z}a_k g(x-k)$；反过来，任意序列$\{a_k\}\in l^2$，确定一个函数$g\in V_0$，且存在正数 A 和 B，且$A\le B$，使得$A\|u\|^2\le\sum_{k\in Z}|a_k|^2\le B\|u\|^2$则称$V_j=\{\varphi_{j,k}(x)\}$为一个多分辨分析。

引入闭子空间$W_j, j\in Z$，构成V_j在V_{j-1}空间的正交补空间，即$V_{j+1}=V_j\oplus W_j, V_j\perp W_j$，$\psi_{j,k}(x)$是$W_j, j\in Z$中的一组标准正交基。它的平移伸缩系为：

$$\psi_{j,k}(x)=2^{j/2}\psi\left(2^{j}x-k\right)$$

由多分辨分析的性质可知，$\varphi(x)$ 与 $\psi(x)$ 之间的关系满足双尺度方程：

$$\phi(x)=\sum_{k\in Z}h_k\phi(2x-k)$$

$$\psi(x)=\sum_{k\in Z}g_k\phi(2x-k)$$

为保证正交性，必须满足条件：$g_k=(-1)^k h_k$

其中 $\varphi(x)$、$\psi(x)$ 被分别称为尺度函数与小波函数。继续将 V_j 空间进行分解，$V_{j_2+1}=V_{j_1}\oplus W_{j_1}\oplus W_{j_1+1}\oplus W_{j_1+2}\oplus\ldots\oplus W_{j_2},(j_1,j_2\in Z;j_1\leq j_2)$，因此 $L^2(R)=\underset{j\in Z}{\oplus}W_j$。对于一个函数 $f(x)\in L^2(R)$，$f(x)$ 在多分辨分析 $\{V_j\}$ 下可以近似的表示为：

$$f(x)\cong A_j f(x)=\sum_{k=-\infty}^{+\infty}C_{j,k}\varphi_{j,k}(x)$$

这里 $C_{j,k}=\langle f(x),\varphi_{j,k}\rangle$。按照定义，每个 $C_{j,k}$ 都要计算尺度函数与 $f(x)$ 的内积，计算量非常大，因此要考虑小波变换的快速算法。

6.4.2 离散小波变换和 Mallat 算法

离散小波变换是对连续小波变换的尺度和位移按照 2 的幂次进行离散化得到的，又称二进制小波变换。离散小波变换可以表示为：

$$W_k[f(x)]=\frac{1}{2^k}\int_{-\infty}^{\infty}f(t)\Psi^*\left(\frac{x-t}{2^k}\right)dt,$$

其中 $\Psi(t)$ 是小波母函数。

实际上，人们是在一定尺度上认识信号的，人的感官和物理仪器都有一定的分辨率，对低于一定尺度的信号的细节是无法认识的，因此对低于一定尺度信号的研究也是没有意义的。为此应该将信号分解为对应不同尺度的近似分量和细节分量。小波分解的意义就在于能够在不同尺度上对信号进行分析，而且对不同尺度的选择可以根据不同的目的来确定。

信号的近似分量一般为信号的低频分量，它的细节分量一般为信号的高频分量，因此对信号的小波分解可以等效于信号通过了一个滤波器组，其中一个滤波器为低通滤波器，另一个为高通滤波器，其示意图如图 6.29 所示。

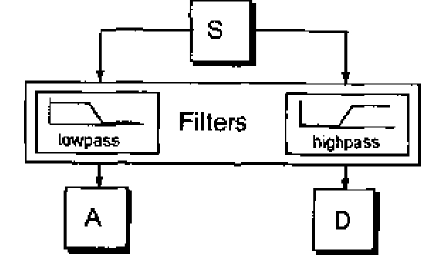


图 6.29 小波分解示意图

以后，用 H 表示高通滤波器，用 L 表示低通滤波器。

Mallat 算法在是小波分解的快速算法，与 FFT 在 Fourier 分析中的作用相似。只有在小波分解的快速算法出现之后，小波分析的实际意义才

为人们所重视。

下面介绍一下 Mallat 算法的实现和小波分解的结构。

设$\{V_j\}$是一个给定的多分辨分析，$\varphi(x)$ 与 $\psi(x)$ 是尺度函数与小波函数函数，$f(x)$在尺度 j 上可以近似的表示为：

$$f(x)\cong A_j f(x)=\sum_{k=-\infty}^{+\infty}C_{j,k}\varphi_{j,k}(x)=A_{j-1}f(x)+D_{j-1}f(x)=\sum_{m=-\infty}^{\infty}C_{j-1,m}\varphi_{j-1,k}(x)+\sum_{m=-\infty}^{\infty}D_{j-1,m}\psi_{j-1,k}(x)$$

这里 $A_{j-1}f(x)$ 表示在第 j−1 尺度上对信号的近似，$D_{j-1}f(x)$ 表示在信号在第 j−1 尺度上的细节。

根据多分辨分析的双尺度方程：$\begin{cases}\langle\varphi_{j,k},\varphi_{j-1,m}\rangle=\tilde{h}_{k-2m}\\ \langle\varphi_{j,k},\psi_{j-1,m}\rangle=\tilde{g}_{k-2m}\end{cases}$

可以求出 $\begin{cases}C_{j-1,m}=\sum\limits_{k=-\infty}^{\infty}\tilde{h}_{k-2m}C_{j,m}\\ D_{j-1,m}=\sum\limits_{k=-\infty}^{\infty}\tilde{g}_{k-2m}C_{j,m}\end{cases}$

引入无穷矩阵 $H=(H_{m,k}),G=(G_{m,k})$，$H_{m,k}=\tilde{h}_{k-2m},G_{m,k}=\tilde{g}_{k-2m}$，则上式的变换关系可以写成下面简单的形式：

$$\begin{cases}C_{j-1}=HC_j\\ D_{j-1}=GC_j\end{cases}$$

一个 1000 点的信号利用小波分解后的结果如图 6.30 所示。

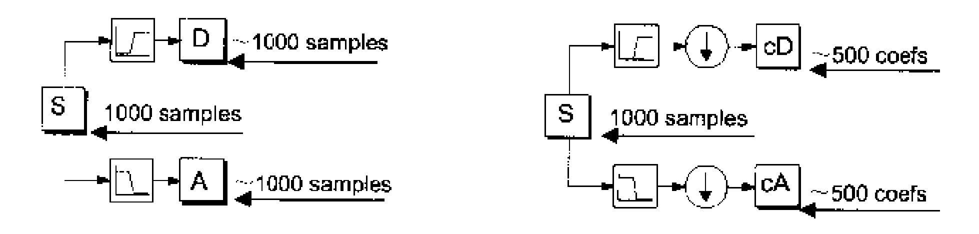


图 6.30 小波分解1000点的信号结果

其中 A 为原始信号的近似信号，D 为细节信号。cA 是小波分解的近似分量的系数，cD 是细节分量的系数。使用中要注意区分近似信号和近似分量、细节信号和细节分量的区别。我们一般称 cA 和 cD 为近似分量和细节分量，而称 A 和 D 为近似信号和细节信号。它们的关系为

$$A(t)=\sum_k cA_k\varphi_k(t)$$

和

$$D(t)=\sum_k cD_k\Psi_k(t)$$

二式是小波的分解算法，我们可以依次逐级分解下去，这样就构成了多重小波分解的

递推形式，如图 6.31 所示。

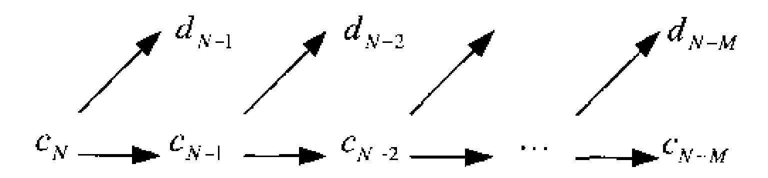


图 6.31 多重小波分解的递推形式

图中的 $c_k = \{c_{k,n}\}, d_k = \{d_{k,n}\}$。

多层小波分解的示意图如图 6.32 所示。

小波分解的意义在于，可以使我们在任意尺度上观察信号，只要使用的小波函数的尺度合适。小波分解将信号分解为分量和细节分量，它们在应用中分别有不同的特点。比如对于含噪信号，噪声分量的主要能量一般集中在小波分解的细节分量中，因此对细节分量进行阈值处理可以滤除噪声。

将信号的小波分解的分量进行处理后，一般根据需要把信号恢复出来，也就是利用信号的小波分解的分量重构出原来的信号或者所需要的信号。小波的重构算法如图 6.33 所示。

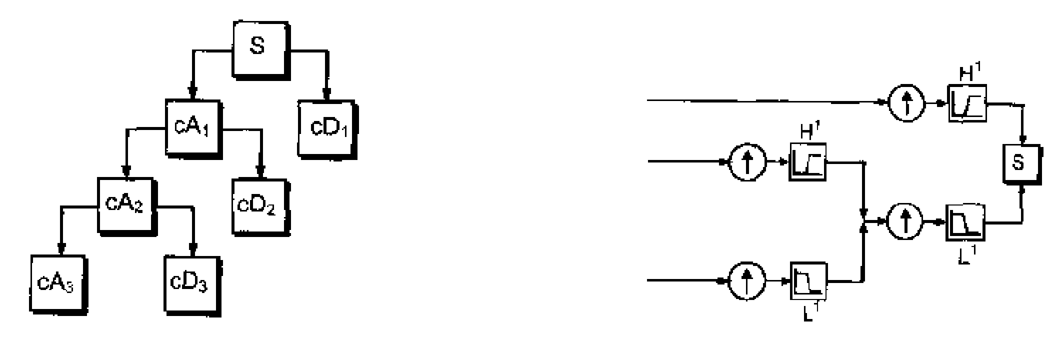


图 6.32 多层小波分解示意图

图 6.33 小波重构算法示意图

对小波分解的式子两边用函数 $\varphi_{j,k}$ 作内积，得到小波的重构算法如下：

$$c_{j,k} = \sum_{m=-\infty}^{\infty} h_{k-2m} c_{j-1,m} + \sum_{m=-\infty}^{\infty} g_{k-2m} d_{j-1,m}$$

用数学式子表示为：

$$C_{j,k} = H^* C_{j-1} + G^* D_{j-1}$$

其中 H^* 和 G^* 分别是 H 和 G 的共轭转置矩阵。

类似的，小波重构也可以推导出多重递推结构，如图 6.34 所示。

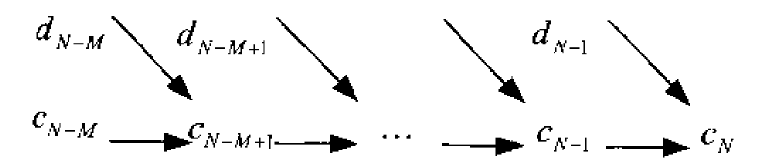


图 6.34 多重小波重构的递推形式

多重小波重构如图 6.35 所示。

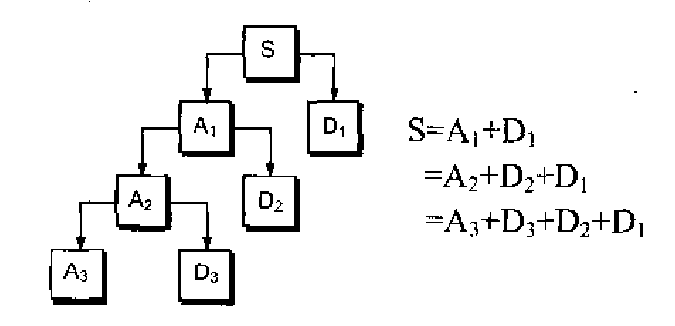


图 6.35 多重小波重构示意图

定义矩阵 $W=\begin{bmatrix}H\\G\end{bmatrix}$，因此小波分解算法可以表示为：

$$\begin{bmatrix}C_{j-1}\\D_{j-1}\end{bmatrix}=WC_j$$

而重构算法可以表示为：$C_j=W^*\begin{bmatrix}C_{j-1}\\D_{j-1}\end{bmatrix}$

信号的多层小波分解和重构可以表示为如图 6.36 所示。

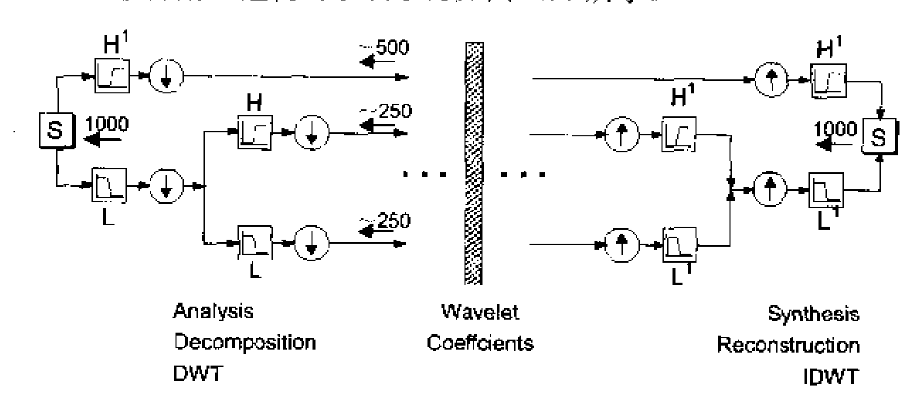


图 6.36 多重小波分解和重构示意图

在小波分解中，一个信号可以不断分解为近似信号和细节信号，近似信号可以继续分解，但是细节信号不能分解。为此，人们又提出了对信号的小波包分解。使用小波包分解，不但可以不断分解近似信号，也可以继续分解细节信号，从而使整个分解构成一种二分树结构，如图 6.37 所示。

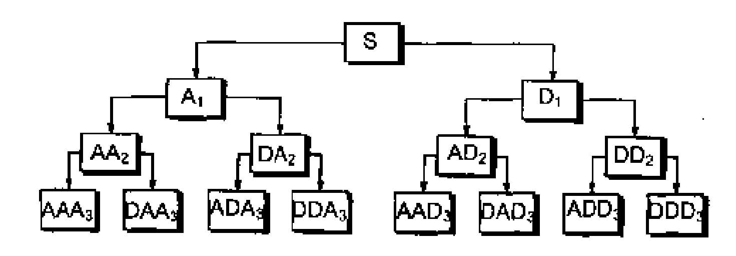


图 6.37 信号的小波包分解

在小波包分解下，一个信号可以有多种表示方式，如图 6.37 所示，信号 S 可以表示为：

```
S=A1+AAD3+DAD3+DD2
```

下面给出小波包的定义：

对于一组正交的小波基函数 $h=\{h_n\},n\in Z$，满足下列条件：

$$\sum h_{n-2k}h_{n-2l}=\delta_{kl},\sum h_n=\sqrt{2}\text{，}$$

令 $g_k=(-1)^k h_{1-k}$，用 $W_0(t)$ 表示多分辨分析的生成元 $\varphi(t)$，$W_1(t)$ 表示小波基函数 $\Psi(t)$，则定义 $\{W_n(t)\},n\in Z$ 为由 $\varphi(t)$ 确定的小波包，其中 $W_n(t)$ 的递归定义为：

$$\begin{cases}W_{2n}(t)=\sqrt{2}\sum h_k W_n(2t-k)\\ W_{2n+1}(t)=\sqrt{2}\sum g_k W_n(2t-k)\end{cases}$$

6.4.3　MATLAB 小波分析工具箱函数介绍

MATLAB 小波分析工具箱提供了很多用于小波分解、重构的函数，下面对重要的小波分析函数进行介绍。

1.　一维离散小波变换函数

- **dwt**

dwt 函数实现一维离散小波变换，其语法格式为：

```
[cA,cD]=dwt(X,'wname')
[cA,cD]=dwt(X,'wname','mode',MODE)
[cA,cD]=dwt(X,Lo_D,Hi_D)
[cA,cD]=dwt(X,Lo_D,Hi_D,'mode',MODE)
```

[cA,cD]=dwt(X,'wname')使用指定的小波基函数'wname'对信号 X 进行小波分解，cA 和 cD 分别为分解得到的近似分量和细节分量。

[cA,cD]=dwt(X,Lo_D,Hi_D)使用指定的滤波器组 Lo_D,Hi_D 对信号进行分解。Lo_D 是低通分解滤波器，用于提取信号的近似分量，Hi-D 是高通分解滤波器，用于提取目标的细节分量。

[cA,cD]=dwt(X,'wname','mode',MODE)和[cA,cD]=dwt(X,Lo_D,Hi_D,'mode',MODE)按照指定的模式对信号进行小波分解。mode 的含义为：

- mode='zpd'：按照边界补零的方式计算小波分解，为默认模式。
- mode='sym'：按照边界缠绕的方式计算小波分解。
- mode='spd'：按照边界平滑的方式计算小波分解。

- **idwt**

idwt 可以实现一维离散小波反变换，其语法格式为：

```
X=idwt(cA,cD,'wname')
```

```
X=idwt(cA,cD,Lo_R,Hi_R)
X=idwt(cA,cD,'wname',L)
X=idwt(cA,cD,Lo_R,Hi_R,L)
X=idwt(…,'mode',MODE)
```

X=idwt(cA,cD, 'wname')是用计算分量 cA 和细节分量 cD 采用小波基函数 wname 进行小波反变换得到的原始信号。X=idwt(cA,cD,Lo_R,Hi_R)用指定的重构滤波器 Lo_R,Hi_R 进行小波反变换得到的原始信号。X=idwt(cA,cD,'wname',L)和 X=idwt(cA,cD,Lo_R,Hi_R,L)是返回经过小波基函数 wname 或者用指定的重构滤波器 Lo_R,Hi_R 小波反变换得到的原始信号中心附近的 L 个点。X=idwt(…,'mode',MODE)是按照指定的计算模式进行小波反变换，mode 的含义与 dwt 函数中的含义相同。

- wavedec

wavedec 用于一维信号的多层小波分解，其语法格式为：

```
[C,L]=wavedec(X,N,'wname')
[C,L]=wavedec(X,N,Lo_D,Hi_D)
```

[C,L]=wavedec(X,N,'wname')使用指定的小波基函数'wname'对信号 X 进行分解，N 指定了分解的层数。[C,L]=wavedec(X,N,Lo_D,Hi_D)对信号进行分解，同样 N 指定了分解的层数。

C 和 L 的意义图如图 6.38 所示。

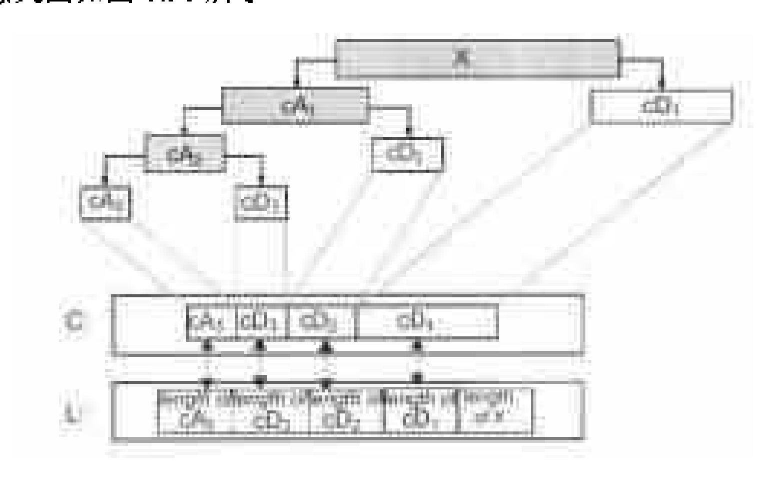


图 6.38 一维小波多层分解的数据结构

也就是说，向量 C 是按照图示的顺序存储信号小波分解的各层的近似分量的系数和细节分量的系数，L 是各近似分量和细节分量系数的长度。

- waverec

waverec 函数用于一维信号的多层重构，与 wavedec 函数互为逆函数，其语法格式为：

```
X=waverec(C,L,'wname')
X=waverec (C,L, Lo_R,Hi_R)
```

X=waverec(C,L,'wname')：利用由 wavedec 函数产生的小波分解[C,L]重构原始信号 X，所用的小波基函数由'wname'决定。X=waverec (C,L, Lo_R,Hi_R)：使用指定的重构滤波器组

Lo_R,Hi_R 重构原始信号 X。

- appcoef

appcoef 函数用来提取一维信号小波分解的近似分量，其语法格式为：

```
A=appcoef(C,L,'wname',N)
A=appcoef(C,L,'wname')
A=appcoef(C,L,Lo_R,Hi_R,N)
A=appcoef(C,L,Lo_R,Hi_R)
```

A=appcoef(C,L,'wname',N) 利用 wavedec 函数产生的多层小波分解结构[C,L]提取第 N 层的近似分量。'wname'为指定的小波基函数的名称，N 必须满足$0 \le N \le length(L)-2$。

A=appcoef(C,L,Lo_R,Hi_R,N) 使用指定的重构滤波器组 Lo_R,Hi-R 提取第 N 层的近似分量。

A=appcoef(C,L, Lo_R,Hi_R,N)和 A=appceof(C,L, Lo_R,Hi_R) 根据 L 的长度来确定提取细节分量的层数，N=length(L)−2。

- detcoef

Detcoef 函数用来提取一维信号小波分解的细节分量，其语法格式为：

```
D=detcoef(C,L,N)
D=detcoef(C,L)
```

D=detcoef(C,L,N)利用 wavedec 函数产生的多层小波分解结构[C,L]提取第 N 层的近似分量。如果不指定 N，N 默认为 length(L)−2。

- upcoef

Upcoef 函数由一维小波分解重构近似分量和细节分量，其语法格式为：

```
Y=upcoef(O,X,'wname',N)
Y=upcoef(O,X,'wname',N,L)
Y=upcoef(O,X,Lo_R,Hi_R, N)
Y=upcoef(O,X,Lo_R,Hi_R, N,L)
Y=upcoef(O,X,'wname')
Y=upcoef(O,X,Lo_R,Hi_R)
```

Y=upcoef(O,X,'wname',N)是由一维离散小波变换系数重构原始信号的近似信号和细节信号，参数 O 指定所要重构的是近似信号还是细节信号，即：

- O='a'，重构近似信号，相应的 X 是原始信号分解的近似分量；
- O='d'，重构细节信号，相应的 X 是原始信号分解的细节分量。

'wname'是使用的小波基函数的名称，N 指定重构的次数。

- wrcoef

wrcoef 实现一维近似信号或者细节信号的小波重构，其格式为：

```
X=wrcoef('type',C,L,'wname',N)
X=wrcoef('type',C,L,Lo_R,Hi_R,N)
```

```
X=wrcoef('type',C,L,'wname')
X=wrcoef('type',C,L,Lo_R,Hi_R )
```

X=wrcoef('type',C,L, 'wname',N)利用 wavedec 函数产生的小波分解结构[C,L]重构第 N 层的近似信号或者细节信号，'wname'为指定的小波基函数的名称。

type 决定重构近似信号还是细节信号：

- Type='a'时，重构近似信号。
- Type='d'时，重构细节信号。

X=wrcoef('type',C,L,Lo_R,Hi_R,N) 使用指定的重构滤波器组 Lo_R,Hi-R 来重构信号，同样 N 指定重构信号的层数，type 决定重构近似信号还是细节信号。

X=wrcoef('type',C,L,'wname') 和 X=wrcoef('type',C,L,Lo_R,Hi_R)根据 L 的长度来确定重构信号的层数，N=length(L)−2。

- upwlev

upwlev 函数用于重构一维小波分解结构，其格式为：

```
[NC,NL,cA]=upwlev(C,L,'wname')
[NC,NL,cA]=upwlev(C,L,Lo_R,Hi_R)
```

[NC,NL,cA]=upwlev(C,L,'wname')利用由 wavedec 函数产生的小波分解结构[C,L]重构上一层的分解结构[NC,NL]。同时还返回细节分解系数 cA。所用的小波基函数由'wname'决定。

[NC,NL,cA]=upwlev(C,L, Lo_R,Hi_R)使用指定的重构滤波器组 Lo_R,Hi_R 重构上一层小波分解结构[NC,NL] 及细节分解系数 cA。

下面以一个例子来讲解上面介绍的各个函数的使用方法和用途。MATLAB 预先存储了一个电源信号 leleccum.mat，比较逼真地表现了信号的起伏以及噪声的影响。下面就按照信号分解、重构的顺序介绍小波分析函数的应用。这里采用 db1 小波。

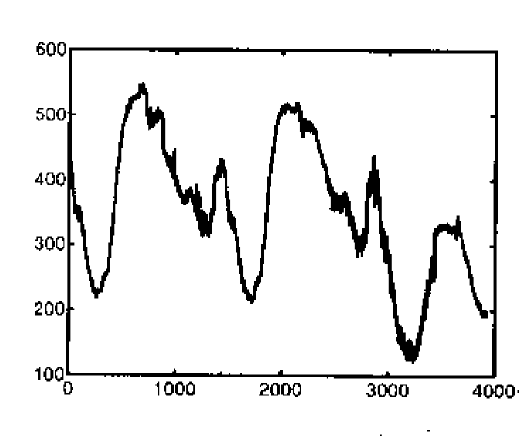


图 6.39 原始电源信号

(1) 首先读入图 6.39 所示的信号。

```
load leleccum
s=leleccum(1:3920);
ls=length(s);
plot(s)
```

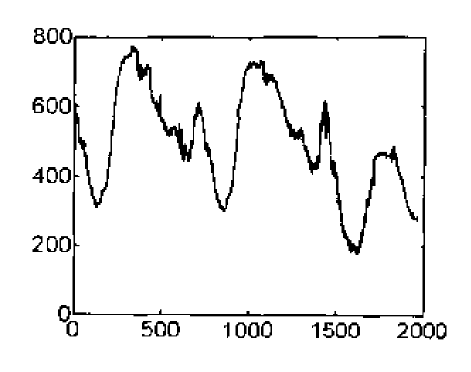
(2) 用 dwt 函数对读入信号进行小波分解，得到第 1 层的近似分量和细节分量，如图 6.40 所示。

```
[ca1,cd1]=dwt(s,'db1');
plot(ca1);
figure,plot(cd1)
```

(3) 利用得到的第 1 层的近似分量和细节分量采用 upcoef 重构近似信号和细节信号，如图 6.41 所示。

```
a1=upcoef('a',ca1,'db1',1,ls);
d1=upcoef('d',cd1,'db1',1,ls);
```

```
plot(a1)
figure,plot(d1)
```



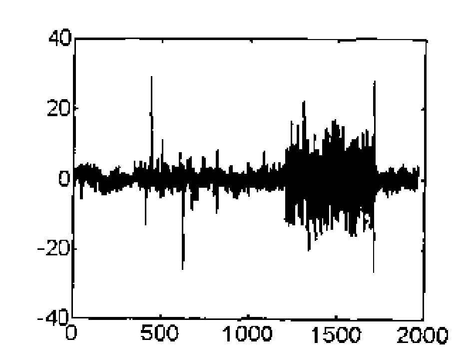
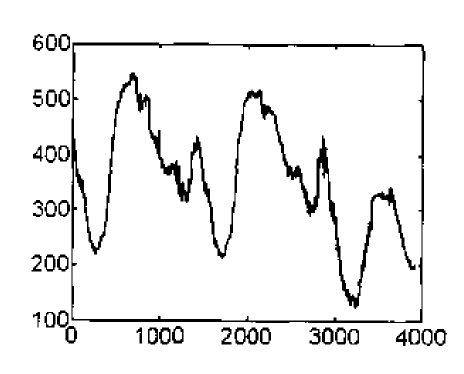


图 6.40 信号小波分解第一层的近似分量和细节分量



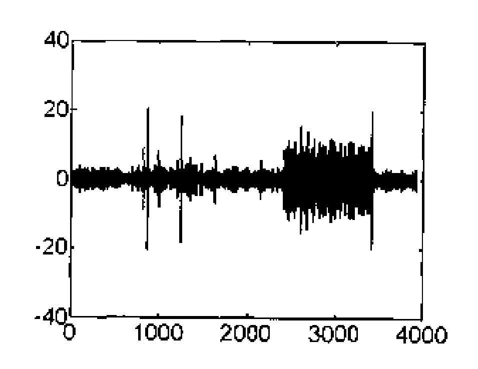


图 6.41 利用近似分量和细节分量重构得到的近似信号和细节信号

(4) 将重构的近似信号和细节信号相加，结果如图 6.42 所示，与原始信号进行比较，可知重构效果很好。

```
figure,plot(a1+d1)
```

(5) 利用 idwt 函数和分解得到的近似分量和细节分量重构原始信号，如图 6.43 所示比较重构效果。

```
a0=idwt(ca1,cd1,'db1',ls);
figure,plot(a0)
```

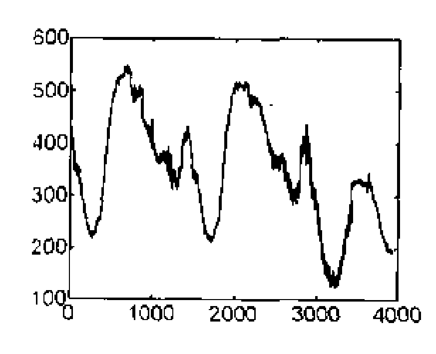


图 6.42 近似信号和细节信号之和

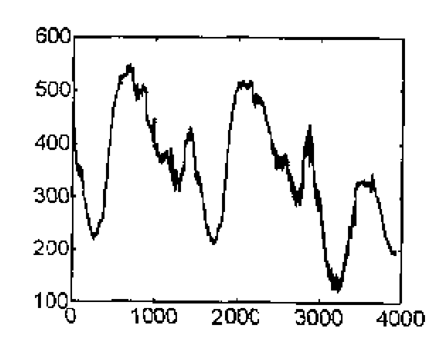


图 6.43 小波反变换重构得到的信号

(6) 利用 wavedec 函数对信号进行第 3 层的小波分解。

```
[c,l]=wavedec(s,3,'db1');
```

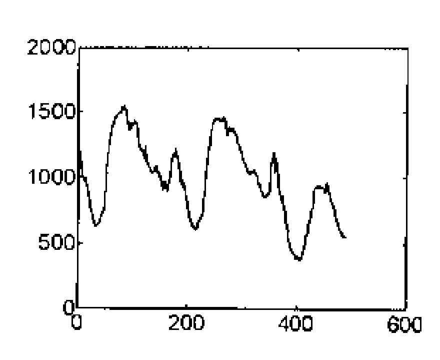
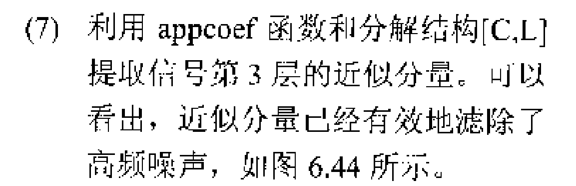


图 6.44 利用小波分解结构提取的信号近似分量

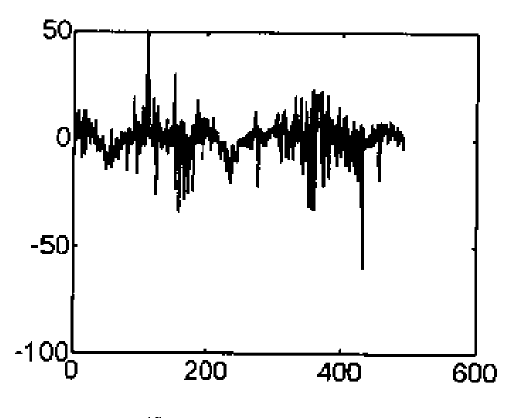


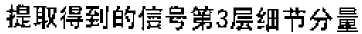
(7) 利用 appcoef 函数和分解结构[C,L]提取信号第 3 层的近似分量。可以看出，近似分量已经有效地滤除了高频噪声，如图 6.44 所示。

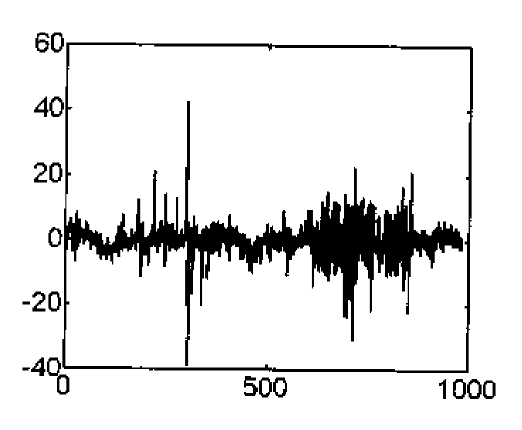
```
ca3=appcoef(c,l,'db1',3);
figure,plot(ca3)
```

(8) 利用 detcoef 函数提取信号第 3、2、1 层的细节分量，如图 6.45 所示。

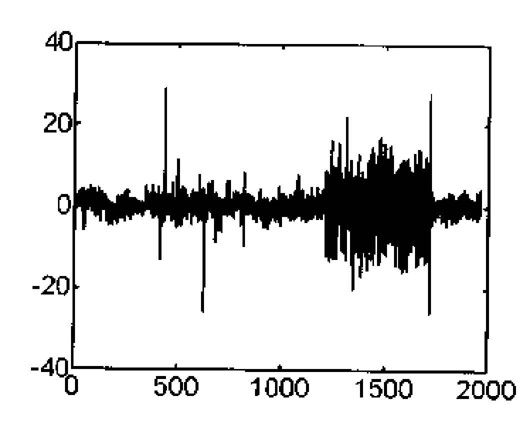
```
cd3=detcoef(c,l,3);
cd2=detcoef(c,l,2);
cd1=detcoef(c,l,1);
plot(cd3);
figure,plot(cd2)
figure,plot(cd1);
```




提取得到的信号第3层细节分量



提取得到的信号第2层细节分量



提取得到的信号第1层细节分量

图 6.45 提取到的信号细节分量

(9) 利用 wrcoef 函数重构信号第 3 层的近似信号，结果如图 6.46 所示。

```
a3=wrcoef('a',c,l,'db1',3);
figure,plot(a3)
```

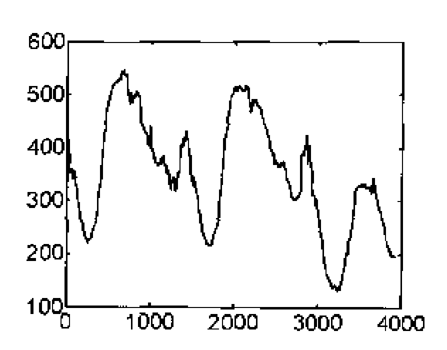
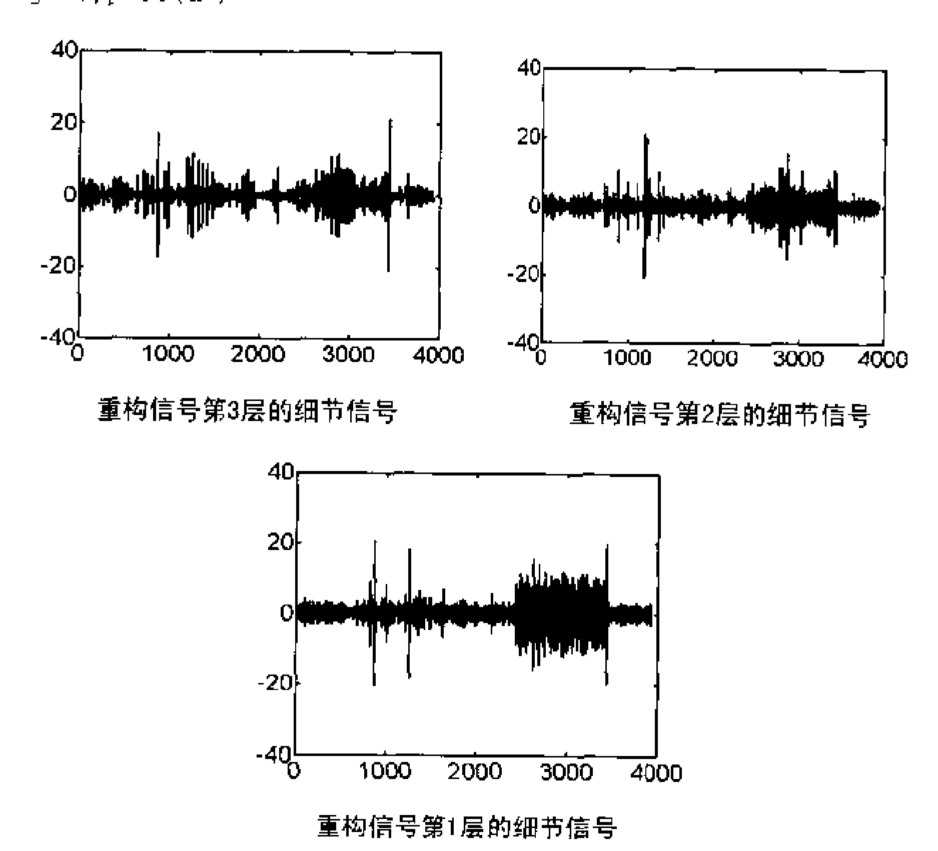


图 6.46　重构信号第3层的近似信号

(10) 利用 wrcoef 函数重构信号第 3、2、1 层的细节信号，如图 6.47 所示。

```
d3=wrcoef('d',c,l,'db1',3);
d2=wrcoef('d',c,l,'db1',2);
d1=wrcoef('d',c,l,'db1',1);
plot(d3);
figure,plot(d2)
figure,plot(d1)
```



重构信号第3层的细节信号

重构信号第2层的细节信号

重构信号第1层的细节信号

图 6.47　重构信号的细节信号

(11) 利用 waverec 函数和分解结构[C,L]重构信号在第 1 层的近似信号，如图 6.48 所示。

```
a0=waverec(c,l,'db1');
figure,plot(a0)
```

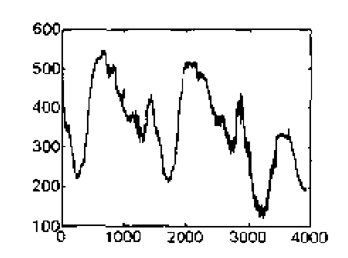


图 6.48　重构信号在第1层的近似信号

2.　二维函数小波变换函数

下面再介绍一下 MATLAB 提供的二维离散小波变换函数。二维离散小波变换是一维离散小波变换的推广，应用张量的概念，很容易推出二维离散小波变换的定义和性质。

设 $\{V_j\}_{j\in Z}$ 是 $L^2(R)$ 空间中的闭子空间列，容易证明，对于张量积空间 $\{V_j^2\}_{j\in Z}$，其中：

$$V_j^2 = V_j \otimes V_j$$

构成 $L^2(R)$ 空间中一个多分辨分析，当且仅当 $\{V_j\}_{j\in Z}$ 是 $L^2(R)$ 空间的一个多分辨分析。并且，二维多分辨分析 $\{V_j\}_{j\in Z}$ 的尺度函数为：

$$\Phi(x,y) = \varphi(x)\varphi(y)\text{。}$$

其中 φ 是一维多分辨分析 $\{V_j\}_{j\in Z}$ 的实值尺度函数。

因此二维离散小波变换也是将二维信号在不同的尺度上进行分解，得到信号的近似分量和细节分量。由于信号是二维的，所以分解也是二维的。分解的结果为：近似分量 cA，水平细节分量 cH、垂直细节分量 cV 以及对角细节分量 cD。同样也可以利用二维小波分解的结果在不同尺度上重构信号。二维小波分解和重构的示意图如图 6.49 所示。

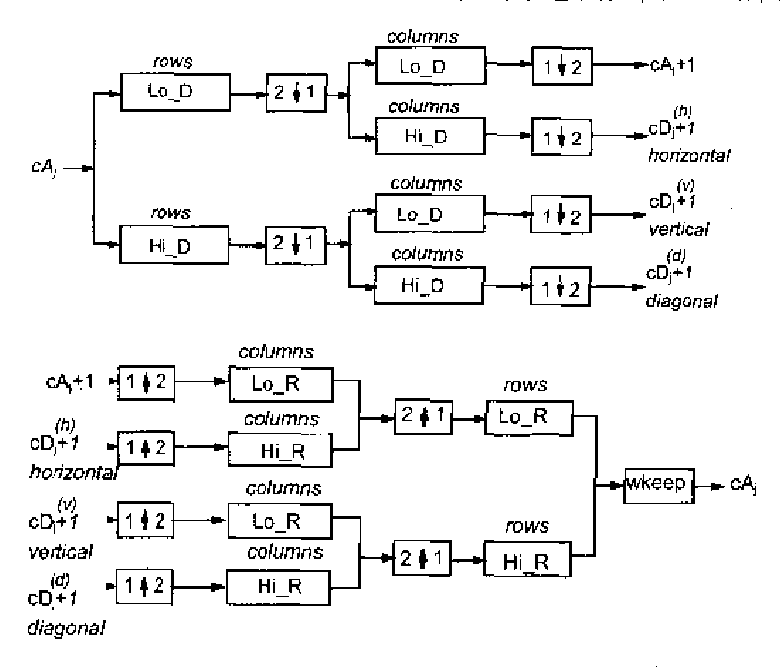


图 6.49　二维小波分解和重构示意图

同样，二维信号也可以进行多层分解和重构，方法与一维离散小波多层分解和重构相似。

MATLAB 提供的二维离散小波变换的函数与一维离散小波变换的函数有着一一对应关系，下面简单介绍。

通常处理的图像很多为索引色图像，图像矩阵各元素表示的是调色板中的序号。而小波分析是对数值进行分析的，因此要将索引色图像进行编码，进行小波分析才有实际意义。MATLAB 提供了 wcodemat 函数来对图像进行编码。

- wcodemat

wcodemat 函数用于对索引色图像的数据矩阵进行编码，其语法格式为：

```
Y=wcodemat(X,NB,OPT,ABSOL)
Y=wcodemat(X,NB,OPT)
Y=wcodemat(X,NB)
Y=wcodemat(X)
```

Y=wcodemat(X,NB,OPT,ABSOL)对索引色图像的数据矩阵 X 进行编码，Y 为编码返回值。NB 是最大编码值，决定了编码范围是 0~NB，默认值为 16。

OPT 指定编码方式，其含义为：

- OPT='row'，对图像按照行进行编码。
- OPT='col'，对图像按照列进行编码。
- OPT='mat'，对图像按照整个矩阵进行编码。

OPT 的默认值为'mat'。

ABSOL 决定返回矩阵的类型：

- ABSOL=0，返回编码矩阵。
- ABSOL=1，返回数据矩阵的绝对值 ABS(X)。

- dwt2

dwt2 函数为二维离散小波变换，语法为：

```
[cA,cH,cV,cD]=dwt2(X,'wname')
[cA,cH,cV,cD]=dwt2(X,Lo_D,Hi_D)
```

[cA,cH,cV,cD]=dwt2(X,'wname')使用指定的小波基函数'wname'对图像 X 进行二维离散小波变换。cA,cH,cV,cD 分别为图像分解的近似分量、水平分量、垂直分量和细节分量。

[cA,cH,cV,cD]=dwt2(X, Lo_D,Hi_D)使用指定的低通和高通滤波器组 Lo_D,Hi_D 来对图像进行二维离散小波变换。

- idwt2

idwt2 函数用于二维离散小波反变换。

```
X=idwt2(cA,cH,cV,cD,'wname')
X=idwt2(cA,cH,cV,cD,'wname',S)
X=idwt2(cA,cH,cV,cD,Lo_R,Hi_R)
```

```
X=idwt2(cA,cH,cV,cD,Lo_R,Hi_R,S)
```

X=idwt2(cA,cH,cV,cD,'wname')利用小波分解得到的 cA,cH,cV,cD 分量进行二维离散小波反变换得到原始图像。'wname'指定二维离散小波反变换采用的小波基函数。

X=idwt2(cA,cH,cV,cD,Lo_R,Hi_R) 利用小波分解得到的 cA,cH,cV,cD 分量进行二维离散小波反变换得到原始图像。Lo_R,Hi_R 为指定的重构滤波器组。

X=idwt2(cA,cH,cV,cD, 'wname',S)和 X=idwt2(cA,cH,cV,cD, Lo_R,Hi_R,S)返回二维离散小波反变换结果的中间附近 S 个点的值。

- wavedec2

wavedec2 函数用来对二维图像进行多层小波分解

```
[C,S]=wavedec2(X,N,'wname')
[C,S]=wavedec2(X,N,Lo_D,Hi_D)
```

[C,S]=wavedec2(X,N, 'wname')使用指定的小波基函数'wname'对图像 X 进行 N 层二维离散小波分解。

[C,S]= wavedec2(X,N, Lo_D,Hi_D) 使用指定的低通和高通滤波器组 Lo_D,Hi_D 来对图像进行 N 层二维离散小波分解。

数据矩阵 C 和长度矩阵 S 的意义如图 6.50 所示。

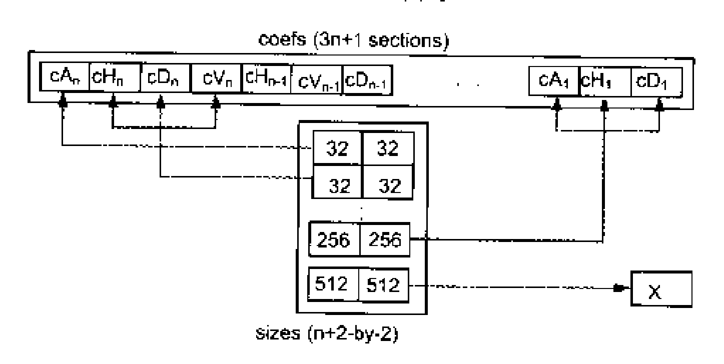


图 6.50 数据矩阵和长度矩阵的意义

- waverec2

waverec2 函数用来进行二维信号的多层小波重构。

```
X=waverec2(C,S,'wname')
X=waverec2(C,S,Lo_R,Hi_R)
```

X=waverec2(C,S,'wname')利用二维小波分解得到的数据矩阵 C 和长度矢量 S 重构原始图像。'wname'为指定的小波基函数。

X=waverec2(C,S, Lo_R,Hi_R)使用重构滤波器组 Lo_R,Hi_R 重构图像。

- appcoef2

appcoef2 函数用来提取二维信号小波分解的近似分量。

```
A=appcoef2(C,S,'wname',N)
```

```
A=appcoef2(C,S,'wname')
A=appcoef2(C,S,Lo_R,Hi_R,N)
A=appcoef2(C,S,Lo_R,Hi_R)
```

A=appcoef2(C,S,'wname',N)利用二维离散小波分解 wavedec2 函数产生的多层小波分解结构 C 和 S 来提取信号第 N 层的近似分量，'wname'为指定的小波基函数名称，N 的默认值为 N=size(S(1,:))−2，即 PN 默认为长度矩阵 S 的行数减去 2。

A=appcoef2(C,S,Lo_R,Hi_R,N) 和 A=appcoef2(C,S,Lo_R,Hi_R) 用指定的重构滤波器 Lo_R, Hi_R 来重构第 N 层的近似分量，其中参数的含义与 A=appcoef2(C,S,'wname',N)中参数的含义相同。

- detcoef2

detcoef2 函数用来提取二维信号小波分解的细节分量。

```
D=detcoef2(O,C,S,N)
```

D=detcoef2(O,C,S,N)利用二维离散小波分解 wavedec2 函数产生的多层小波分解结构 C 和 S 来提取信号第 N 层的细节分量，O 指定细节信号的类型：

- ◆ O='d'，重构对角细节信号。
- ◆ O='h'，重构水平细节信号。
- ◆ O='v'，重构垂直细节信号。

- wrcoef2

wrcoef2 函数由多层二维小波分解来重构某一层的分解信号。

```
X=wrcoef2('type',C,S,'wname')
X=wrcoef2('type',C,S,Lo_R,Hi_R)
X=wrcoef2('type',C,S,'wname',N)
X=wrcoef2('type',C,S,Lo_R,Hi_R,N)
```

X=wrcoef2('type',C,S,'wname',N)和 X=wrcoef2('type',C,S,Lo_R,Hi_R,N)用多层小波分解得到的 C 和 S 重构第 N 层的分解信号，N 不指定时，采用 N=size(S,1)−2。

经过重构，返回信号与原始信号大小相同。

'wname'为指定的小波基函数，Lo_R,Hi_R 是指定的。

type 决定要重构分量的类型，含义为：

- ◆ Type='a'，重构近似分量。
- ◆ Type='h'，重构水平分量。
- ◆ Type='v'，重构垂直分量。
- ◆ Type='d'，重构细节分量。

- upcoef2

upcoef2 函数用于利用多层小波分解重构近似分量或者细节分量。

```
Y=upcoef2(O,X,'wname',N)
Y=upcoef2(O,X,Lo_R,Hi_R ,N)
```

```
Y=upcoef2(O,X,'wname',N,S)
Y=upcoef2(O,X,Lo_R,Hi_R ,N,S)
Y=upcoef2(O,X,'wname')
Y=upcoef2(O,X,Lo_R,Hi_R )
```

Y=upcoef2(O,X,'wname',N)由二维离散小波变换得到的系数重构近似分量或者细节分量，'wname'为指定的小波基函数的名称，N 指定重构的次数，它必须是一正整数。若不指定，则 N=1。

O 指定细节信号的类型：

- O='a'，重构近似信号，即 X 是第 N 层的近似系数。
- O='h'，重构水平细节信号，即 X 是第 N 层的水平细节系数。
- O='v'，重构垂直细节信号，即 X 是第 N 层的垂直细节系数。
- O='d'，重构对角细节信号。即 X 是第 N 层的对角细节系数。

Y=upcoef2(O,X, Lo_R,Hi_R ,N)利用指定的重构滤波器组 Lo_R,Hi_R 重构第 N 层的近似分量或者细节分量。

- upwlev2

upwlev2 函数实现二维信号小波分解的单层重构。

```
[NC,NS,cA]=upwlev2(C,S,'wname')
[NC,NS,cA]=upwlev2(C,S,Lo_R,Hi_R)
```

[NC,NS,cA]=upwlev2(C,S, 'wname')利用由 wavedec2 产生的多层小波分解结构 C，S 来重构上一层的分解结构 NC，NS，并返回上一层的近似分量 cA。

[NC,NS,cA]=upwlev2(C,S,Lo_R,Hi_R) 用指定的重构滤波器 Lo_R,Hi_R 来重构上一层的分解结构 NC,NS，并返回上一层的近似分量 cA。

下面也以一个图像为例，说明上面各函数的使用方法。

MATLAB 提供了一幅预存的图像文件 woman2.mat，里面包含一个数据矩阵 X 和一个调色板 map。这里按照图像分解和重构的顺序介绍各函数，采用的小波基函数为'db1'小波。

(1) 读入图像数据，并对其进行编码，结果如图 6.51 所示。

```
load woman2
nbcol=size(map,1);
colormap(pink(nbcol));
cod_X=wcodemat(X,nbcol);
image(cod_X)
axis('square')
```

(2) 对图像进行小波分解，得到近似分量和细节分量，如图 6.52 所示。

```
[ca1,ch1,cv1,cd1]=dwt2(X,'db1');
cod_ca1=wcodemat(ca1,nbcol);
cod_ch1=wcodemat(ch1,nbcol);
cod_cv1=wcodemat(cv1,nbcol);
cod_cd1=wcodemat(cd1,nbcol);
image([cod_ca1,cod_ch1;cod_cv1,cod_cd1]);
```

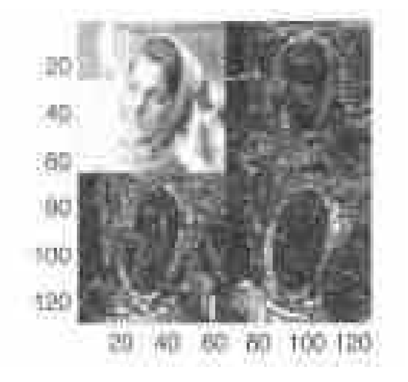


图 6.51　编码后的原始图像

图 6.52　小波分解的近似分量和细节分量

(3) 对图像进行二次分解，得到第 2 层的近似分量和细节分量，如图 6.53 所示。

```
[ca2,ch2,cv2,cd2]=dwt2(ca1,'db1');
cod_ca2=wcodemat(ca2,nbcol);
cod_ch2=wcodemat(ch2,nbcol);
cod_cv2=wcodemat(cv2,nbcol);
cod_cd2=wcodemat(cd2,nbcol);
image([cod_ca2,cod_ch2;cod_cv2,cod_cd2]);
axis('square')
```

(4) 利用 idwt2 函数在第 1 层重构近似信号，如图 6.54 所示。

```
a0=idwt2(ca1,ch1,cv1,cd1,'db1',size(X));
a0=wcodemat(a0,nbcol);
image(a0)
axis('square')
```

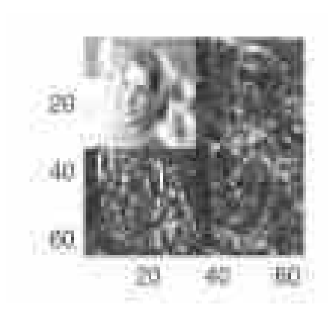


图 6.53　第2层的近似分量和细节分量

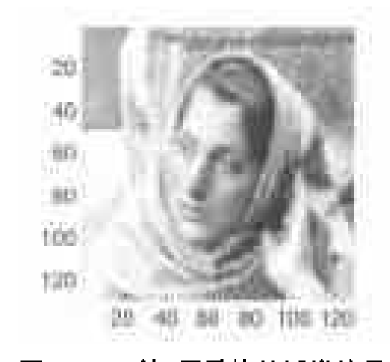


图 6.54　第1层重构的近似信号

(5) 利用 wavedec2 函数在第 2 层对图像进行分解，得到其分解结构[C,S]。

```
[c,s]=wavedec2(X,2,'db1');
```

(6) 利用函数 appcoef2 和分解得到的结构[C,S]提取图像第 2 层和第 1 层的近似分量和细节分量，结果分别如图 6.55 和图 6.56 所示。

```
ca2=appcoef2(c,s,'db1',2);
ch2=detcoef2('h',c,s,2);
cv2=detcoef2('v',c,s,2);
cd2=detcoef2('d',c,s,2);
cod_ca2=wcodemat(ca2,nbcol);
```

```
cod_ch2=wcodemat(ch2,nbcol);
cod_cv2=wcodemat(cv2,nbcol);
cod_cd2=wcodemat(cd2,nbcol);
image([cod_ca2,cod_ch2;cod_cv2,cod_cd2])
axis('square')
ca1=appcoef2(c,s,'db1',1);
ch1=detcoef2('h',c,s,1);
cv1=detcoef2('v',c,s,1);
cd1=detcoef2('d',c,s,1);
cod_ca1=wcodemat(ca1,nbcol);
cod_ch1=wcodemat(ch1,nbcol);
cod_cv1=wcodemat(cv1,nbcol);
cod_cd1=wcodemat(cd1,nbcol);
image([cod_ca1,cod_ch1;cod_cv1,cod_cd1])
axis('square')
```

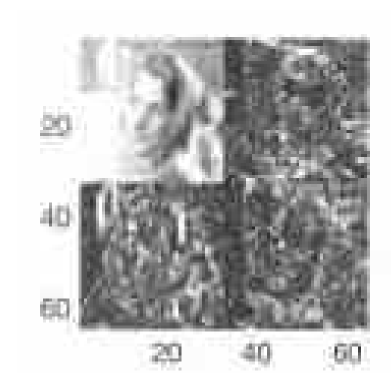


图 6.55 第2层的近似分量和细节分量

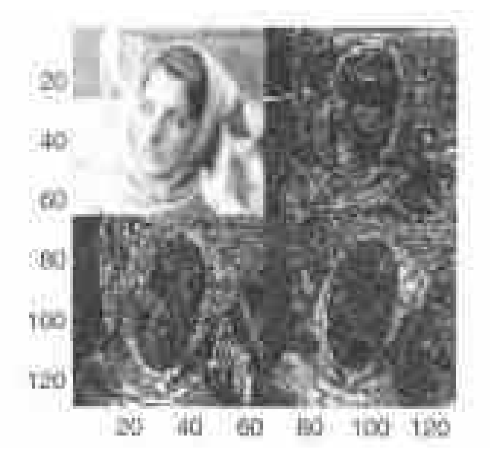


图 6.56 第1层的近似分量和细节分量



图 6.57 第2层重构得到的近似信号

(7) 利用 wrcoef2 函数在图像分解的第 2 层重构近似信号以及细节信号，结果如图 6.57 和图 6.58 所示。

```
a2=wrcoef2('a',c,s,'db1',2);
cod_a2=wcodemat(a2,nbcol);
image(cod_a2)
axis('square')
h2 = wrcoef2('h',c,s,'db1',2);
v2 = wrcoef2('v',c,s,'db1',2);
d2 = wrcoef2('d',c,s,'db1',2);
cod_h2 = wcodemat(h2,nbcol);
cod_v2 = wcodemat(v2,nbcol);
cod_d2 = wcodemat(d2,nbcol);
image(cod_h2)
axis('square')
image(cod_v2)
axis('square')
image(cod_d2)
axis('square')
```

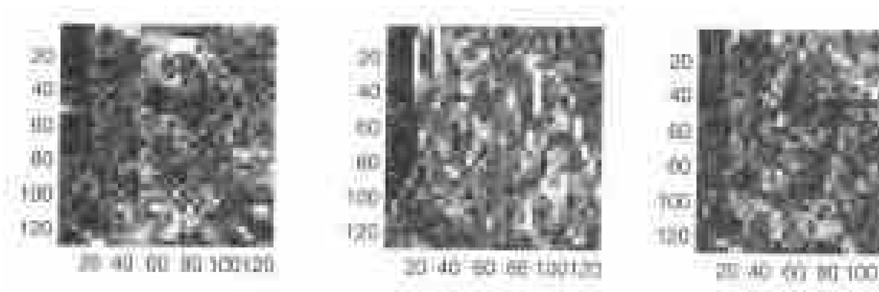


图 6.58 第2层重构得到的细节信号

(8) 利用 upwlev2 来提取图像分解第 1 层的分解结构[C,S]。

```
[c,s]=upwlev2(c,s,'db1');
```

(9) 从分解得到的近似分量和细节分量的系数中重构近似信号和细节信号，要分为两步进行，最后的结果如图 6.59 所示。

① 从分解结构[C,S]中提取系数。

```
ca1=appcoef2(c,s,'db1',1);
ch1=detcoef2('h',c,s,1);
cv1=detcoef2('v',c,s,1);
cd1=detcoef2('d',c,s,1);
```

② 重构。

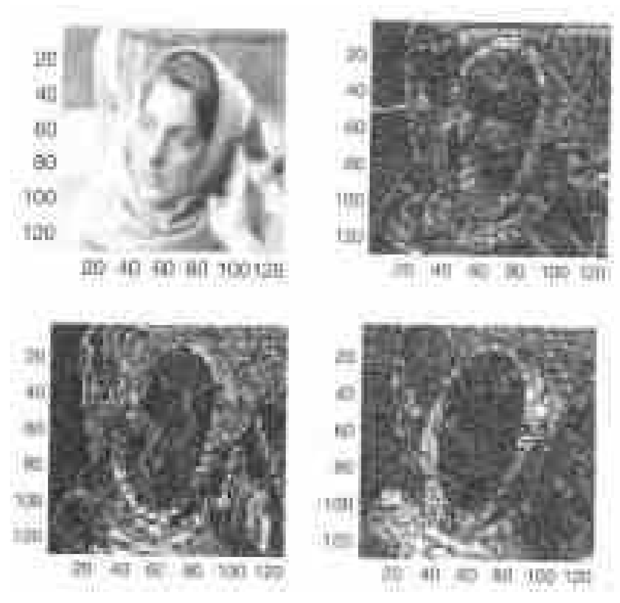


图 6.59 利用近似分量和细节分量的系数重构近似信号和细节信号

```
siz=s(size(s,1),:);
a1=upcoef2('a',ca1,'db1',1,siz);
hd1=upcoef2('h',ch1,'db1',1,siz);
vd1=upcoef2('v',cv1,'db1',1,siz);
dd1=upcoef2('d',cd1,'db1',1,siz);
cod_a1=wcodemat(a1,nbcol);
cod_hd1=wcodemat(hd1,nbcol);
```

```
cod_vd1=wcodemat(vd1,nbcol);
cod_dd1=wcodemat(dd1,nbcol);
image(cod_a1)
axis('square')
image(cod_hd1)
axis('square')
image(cod_vd1)
axis('square')
image(cod_dd1)
axis('square')
```

(10) 从分解结构[C,S]中重构原始信号的近似分量，结果如图 6.60 所示。

```
a0=waverec2(c,s,'db1');
cod_a0=wcodemat(a0,nbcol);
image(cod_a0)
axis('square')
```



图 6.60 利用分解结构重构原始信号的近似分量

第7章 图像增强

图像增强是数字图像处理过程中经常采用的一种方法。为了改善视觉效果或者便于人和机器对图像的理解和分析，根据图像的特点或存在的问题采取的简单改善方法或者加强特征的措施称为图像增强(image enhancement)。

获取和传输图像的过程往往会发生图像失真，所得到图像和原始图像有某种程度的差别。这种差异如果太大，就会影响人和机器对图像的理解。在许多情况下，人们不清楚引起图像降质的具体物理过程及其数学模型，但却能根据经验估计出使图像降质的一些可能原因，针对这些原因采取简便有效的方法，改善图像质量。例如，图像信号变弱会使人们无法看清图像的细节，而采用增强对比度的方法可使图像清晰一些；图像的噪声干扰也容易使图像质量变差，运用平滑技术可以消减噪声；还有一些物理器件或系统工作原理可等效为一积分过程，信号经过这样的器件或系统后要变模糊，这时可使用微分运算突出边界或其他变化的部分。以上这些例子都使用了图像增强的技术。

在有些情况下，一幅图像和其真实景物相比差别不大，人的视觉系统察觉不出来或在容许范围之内，但为了便于人或机器对图像的分析和理解，也需要对图像进行变换来加强图像的某些特征，例如用微分技术加强区域边界。再比如人眼对彩色图像的分辨率要强于灰度图像，因此可以对图像进行伪彩色变换来提高分辨率等。

7.1 直方图增强

7.1.1 直方图

图像的直方图是图像的重要统计特征，它可以认为是图像灰度密度函数的近似。按照随机过程理论，图像可以看作是一个随机场，因此具有相应的统计特征，其中最重要的特征是灰度密度函数。通常图像的灰度密度函数与象素所在的位置有关，例如设图像在点(x,y)处的灰度分布密度函数为p(z;x,y)，那么图像的灰度密度函数为：

$$p(z)=\frac{1}{S}\iint_D p(z;x,y)dxdy$$

其中D是图像的定义域，S是区域D的面积。一般地讲，要精确地得到图像的灰度密度函数是比较困难的，在实际中可以用数字图像灰度直方图来代替。灰度直方图是一个离散函数，它表示数字图像每一灰度级与该灰度级出现频率的对应关系。假设一幅数字图像的像素总数为N，有L个灰度级，具有第k个灰度级的灰度 r_k 的像素共有 n_k 个。则第k

个灰度级或者说 r_k 出现的频率为：

$$h_k = \frac{n_k}{N} \qquad k=0,1,\ldots,L-1$$

这个关系也可以用图形表示。这个图形是由灰度轴及一系列垂直于灰度轴的线段组成，垂足 $f=r_k$，各线段长度与 h_k 成正比。

MATLAB 图像处理工具箱提供了 imhist 函数来计算和显示图像的直方图，imhist 函数的语法格式为：

```
imhist(I,n)
imhist(X,map)
[counts,x]=imhist(…)
```

其中 imhist(I,n)计算和显示灰度图像 I 的直方图，n 为指定的灰度级数目，默认值为256。imhist(X,map)计算和显示索引色图像 X 的直方图，map 为调色板。[counts,x]=imhist(…)返回直方图数据向量 counts 或相应的色彩值向量 x。

图像 rice.tif 的灰度直方图如图 7.1 所示。

```
I=imread('rice.tif')
imshow(I)
figure,imhist(I)
```

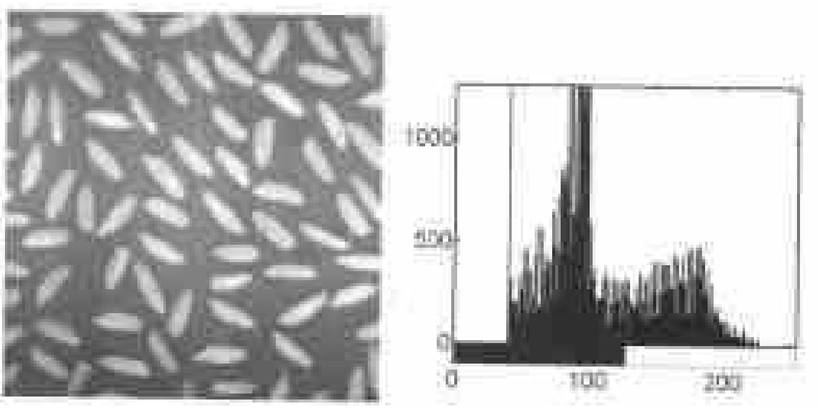


图 7.1　灰度图像及其直方图

密度函数 p(z)或直方图 h_k 虽然不能直接反映出图像内容，但对它进行分析可以得出图像的一些有用特征，这些特征能反映出图像的特点。例如，当图像对比度较小时，它的灰度直方图只在灰度轴上较小的一段区间上非零，较暗的图像由于较多的像素灰度值低，因此它的直方图的主体出现在低值灰度区间上，其在高值灰度区间上的幅度较小或为零，而较亮的图像情况正好相反。看起来清晰柔和的图像，它的直方图灰度分布比较均匀。通常一幅均匀量化的自然图像的灰度直方图在低值灰度区间上频率较大，这样的图像较暗区域中的细节常常看不清楚。为使图像变清晰，通常可以通过变换使图像的灰度动态范围变大，并且让灰度频率较小的灰度级经变换后，其频率变得大一些，使变换后的图像灰度直方图在较大的动态范围内趋于均化。事实证明，通过图像直方图修改进行图像增强是一种有效的方法。另外还有一种图像增强技术，它是以直方图作为变换的依据，使变换后的图像直方图成为期望的形状。这样也可以提高图像的对比度，而且可以根据需要增强感兴趣的灰

度范围。

7.1.2　直方图均化

均匀量化的自然图像的灰度直方图通常在低值灰度区间上频率较大，使得图像中较暗区域中的细节常常看不清楚。为了使图像清晰，可将图像的灰度范围拉开，并且让灰度频率较小的灰度级变大，即让灰度直方图在较大的动态范围内趋于一致。

用图像 f(x,y)的直方图代替灰度的分布密度函数 $p_f(f)$，则直方图均化后的图像 g 为：

$$g = T[f] = \int_0^f p_f(u)du$$

对于数字图像，可以对上述公式做离散近似。设原图像的像素总数为 N，灰度级的个数为 L，第 k 个灰度级出现的频数为 n_k。若原图像 f(x,y)在像素点(x,y)处的灰度为 r_k，则直方图均化化后的图像 g(x,y)在处的灰度 s_k 为：

$$s_k = T[r_k] = \sum_{l=0}^{k} \frac{n_l}{N}$$

MATLAB 图像处理工具箱提供了用于直方图均化的函数 histeq。

histeq 函数的语法为：

```
J=histeq(I,hgram)
J=histeq(I,n)
[J,T]=histeq(I,…)
newmap=histeq(X,map,hgram)
newmap=histeq(X,map)
[newmap,T]=histeq(X,…)
```

其中 J=histeq(I,hgram)将原始图像 I 的直方图变成用户指定的向量 hgram，hgram 中的各元素值域为[0,1]。

J=histeq(I,n)指定直方图均化后的灰度级数 n，默认值为 64。

[J,T]=histeq(I,…)返回从能将图像 I 的灰度直方图变换成图像 J 的直方图的变换 T。

newmap=histeq(X,map,hgram)、newmap=histeq(X,map)和[newmap,T] =histeq(X,…)是针对索引色图像调色板的直方图均化。

下例是对图像 tire.tif 作直方图均化，结果如图 7.2 所示。

```
I=imread('tire.tif');
J=histeq(I)
subplot(1,2,1),imshow(I)
subplot(1,2,2),imshow(J)
figure,subplot(1,2,1),imhist(I,64)
subplot(1,2,2),imhist(J,64)
```

经过直方图均化，可以看出，图像的细节更加清楚了，而从图像的直方图上也可以看出，在直方图调整之前，低灰度的比例很大，经过直方图调整，各灰度等级的比例更加平

衡。但是由于直方图均衡没有考虑图像的内容，简单地将图像进行直方图均衡，使得图像看起来亮度过高。也就是说直方图均衡的方法不够灵活，因此人们又提出了别的增强方法。

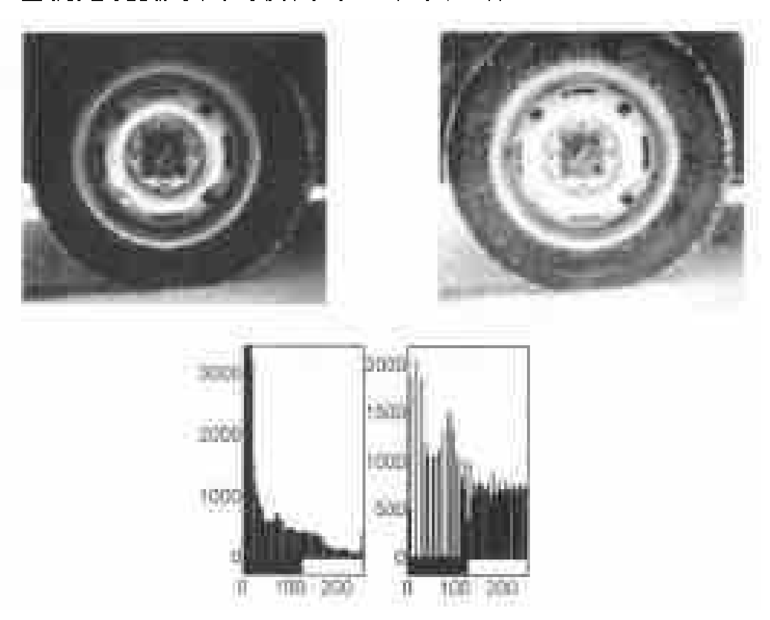


图 7.2 进行直方图均化操作前后的图像及其直方图

7.2 对比度增强

对比度增强是图像增强技术中一种比较简单但又十分重要的方法。这种方法是按一定的规则修改输入图像每一个像素的灰度，从而改变图像灰度的动态范围。它可以使灰度动态范围扩展，也可以使其压缩，或者是对灰度进行分段处理，根据图像特点和要求在某段区间中进行压缩而在另外区间中进行扩展。

例如，观察图 7.3，可以发现，该图像的对比度不高，其灰度直方图没有低于 50 或高于 220 的值，如果将图像数据映射到整个灰度范围内，则图像的对比度将大大增强。

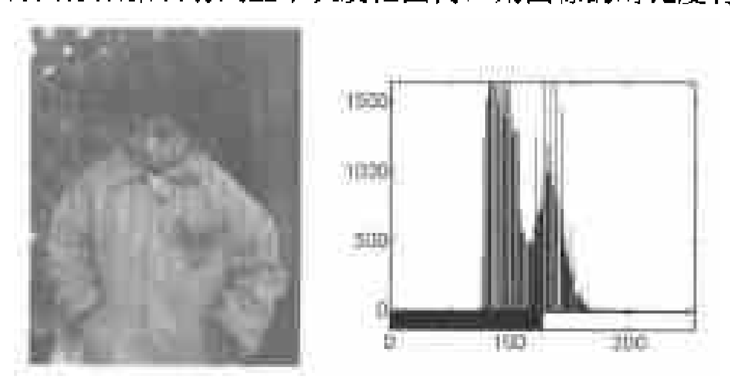


图 7.3 原始图像及其图像直方图

设输入图像为 f(x,y)，处理后的图像为 g(x,y)，则对比度增强可以表示为下面的数学变换式：

$$g(X,Y)=T[f(X,Y)]$$

其中 T 表示输入图像和输出图像对应点的灰度映射关系。实际中由于曝光不足或成像系统非线性的影响，通常照片或电子系统生成的图像对比度不良，利用对比度增强变换可以有效地改善图像的质量。

7.2.1 灰度调整

如果原图像 f(x,y)的灰度范围是[m,M]，我们希望调整后的图像 g(x,y)的灰度范围是[n,N]，那么下述变换，

$$g(x,y)=\frac{N-n}{M-m}[f(x,y)-m]+n$$

就可以实现这一要求。图 7.4 是其变换关系曲线图。

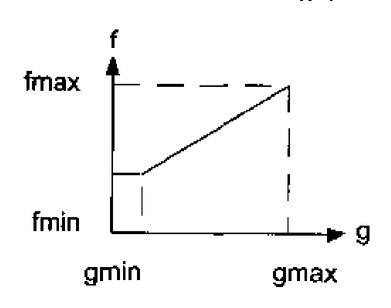


图 7.4 灰度线性变换曲线

MATLAB 图像处理工具箱中提供的 imadjust 函数，可以实现上述的线性变换对比度增强。

1. imadjust函数

imadjust 函数的语法格式为：

```
J=imadjust(I,[low high],[bottom top],gamma)
newmap=imadjust(map,[low high],[bottom top],gamma)
```

J=imadjust(I,[low high],[bottom top],gamma)返回图像 I 经过直方图调整后的图像 J。gamma 为矫正量。[low high]为原图像中要变换的灰度范围，[bottom top]指定了变换后的灰度范围。

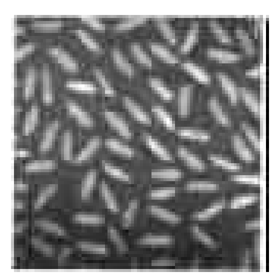
newmap=imadjust(map,[low high],[bottom top],gamma)调整索引色图像的调色板 map。此时若[low high] 和[bottom top]都是 2×3 矩阵，则根据它们的值分别调整 R、G、B3 个分量。

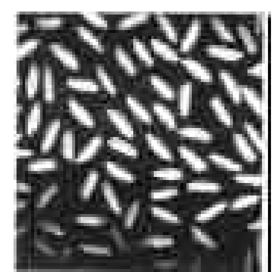
例如调整图像的对比度，调整前后的图像和直方图如图 7.5 所示。

```
I=imread('rice.tif')
J=imadjust(I,[0.3 0.7],[]);
subplot(1,2,1),imshow(I)
subplot(1,2,2),imshow(J)
figure,subplot(1,2,1),imhist(I)
subplot(1,2,2), imhist(J)
```

从原理上讲，我们也可以用一些数学上的非线性函数进行变换，如平方、指数、对数

等，但其中有实际意义的还是对数变换。





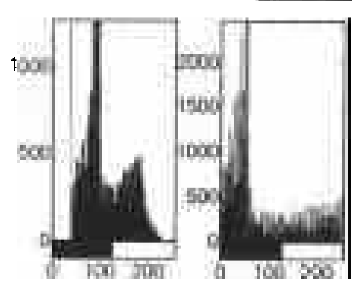


图 7.5 对比度调整前后的图像及其直方图

2. 对数变换

对数变换常用来扩展低值灰度，压缩高值灰度，这样可以使低值灰度的图像细节更容易看清。对数变换的表达式为：

```
g(x,y)=log[f(x,y)+1]
```

运用对数变换的结果如图 7.6 所示，命令如下：

```
I=imread('pout.tif');
Imshow(I)
I=double(I);
%对数运算不支持 uint8 类型数据。
%所以将图像矩阵转化为 double 类型。
J=log(I+1);
figure,imshow(J)
```





图 7.6 对数变换前后图像

从图像对数变换前后的效果比较，可以知道，对数变换确实能够扩展低值灰度，而压缩高值灰度，使低值灰度的图像细节更容易看清。

3. 指数变换

指数变换可以扩展低值灰度，压缩高值灰度，也可以扩展高值灰度，压缩低值灰度，

但是由于与人的视觉特性不太相同，因此不常采用。

7.2.2 Gamma 校正

Gamma 校正也是数字图像处理中常用的图像增强技术。imadjust 函数中的 gamma 因子即是这里所说的 Gamma 校正的参数。Gamma 因子的取值决定了输入图像到输出图像的灰度映射方式，即决定了增强低灰度还是增强高灰度。当 gamma 等于 1 时，为线性变换。

Gamma 因子大于 1 和小于 1 的映射方式如图 7.7 所示。

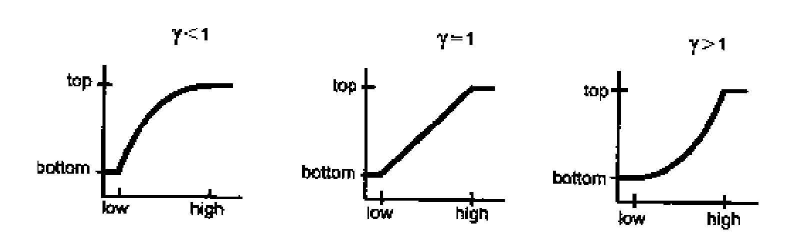


图 7.7 Gamma校正曲线

MATLAB 的演示函数 imadjdemo 分别演示了 gamma 因子取不同值时对输出图像的影响，如图 7.8 和图 7.9 所示。

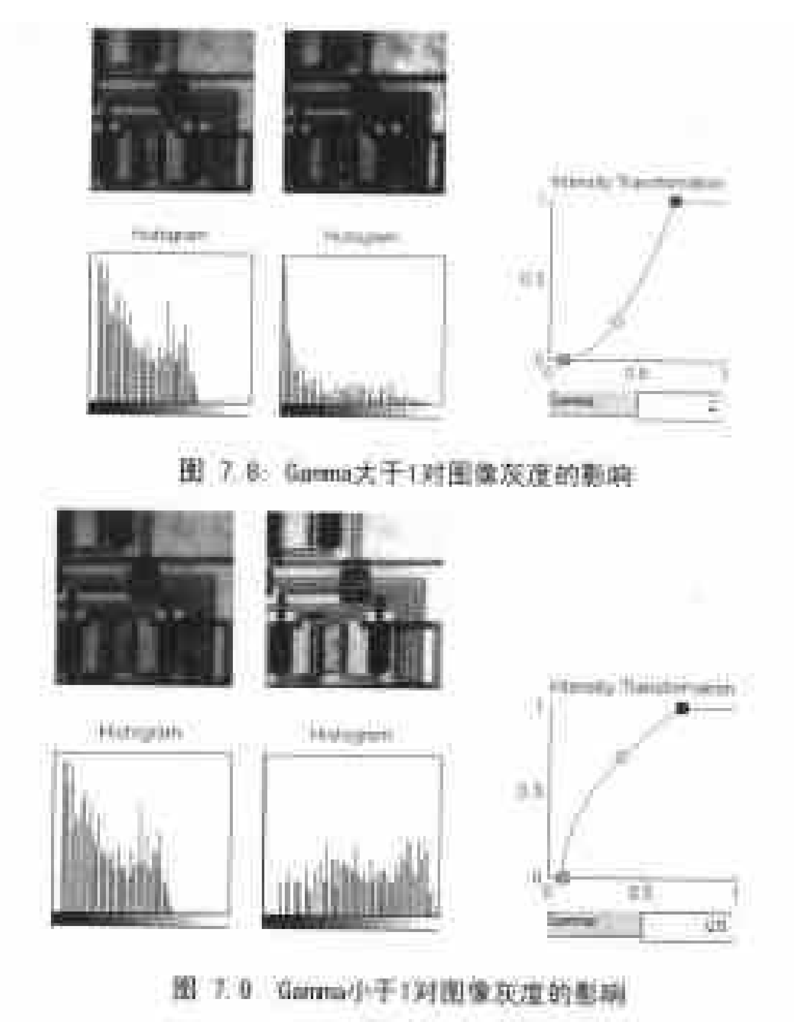


图 7.8 Gamma大于1对图像灰度的影响

图 7.9 Gamma小于1对图像灰度的影响

7.3 二维卷积和二维滤波

线性滤波是一种常用的图像增强技术，可以用来对图像的某些特征进行增强，而去除其他的特征。而且，线性滤波是一种邻域操作，即输出图像的某一像素值由输入图像相应的像素及其邻域的像素的值决定。

由线性系统理论可知，二维系统的输出是系统冲激响应与输入的卷积。二维卷积的定义为：

$$y(m,n)=\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}h(i,j)g(m-i,n-j)$$

MATLAB 图像处理工具箱提供了用于二维和多维卷积的函数。

1. conv2

conv2 函数用于计算二维卷积，其语法格式为：

```
C=conv2(A,B)
C=conv2(...,shape)
```

C=conv2(A,B)返回矩阵 A 和 B 的二维卷积 C。若 A 为 ma×na 的矩阵，B 为 mb×nb 的矩阵，则 C 的大小为(ma+mb+1)×(na+nb+1)。

C=conv2(...,shape)指定卷积运算的范围。

- shape='full'：返回矩阵 C 的大小为(ma+mb+1)×(na+nb+1)。
- shape='same'：返回矩阵 C 的大小与矩阵 A 的大小相同。
- shape='valid'：不考虑边界补零，返回矩阵大小为(ma－mb+1)×(na－nb+1)(假设 A 比 B 大)。

举例如下：

```
A=magic(5)
A =
 17     24      1      8     15
 23      5      7     14     16
  4      6     13     20     22
 10     12     19     21      3
 11     18     25      2      9
B=[1 2 1;0 2 0;3 1 3]
B =
     1     2     1
     0     2     0
     3     1     3
C=conv2(A,B)
C =
  17     58     66     34     32     38     15
  23     85     88     35     67     76     16
  55    149    117    163    159    135     67
  79     78    160    161    187    129     51
```

```
    23      82     153     199     205     108      75
    30      68     135     168      91      84       9
    33      65     126      85     104      15      27
```

2. convmtx2

convmtx2 函数用于计算二维卷积矩阵，其语法格式为：

```
T=convmtx2(H,m,n)
T=convmtx2(H,[m,n])
```

T=convmtx2(H,m,n) 或 T=convmtx2(H,[m,n])返回矩阵 H 的二维卷积矩阵 T,m,n 为待卷积矩阵 X 的尺寸，表示为[m,n]=size(x),conv(X,H)等价于：

```
reshape(T*X(:),size(H)+[m,n]-1)
```

3. convn

convn 函数用于计算 n 维卷积，其语法格式为：

```
convn(A,B)
convn(A,B,shape)
```

其中 convn(A,B)返回多维矩阵 A、B 的 n 维卷积 C，convn(A,B,shape)指定卷积运算的范围：

- shape='full'：作边界补零。
- shape='same'：返回矩阵 C 的大小与矩阵 A 的大小相同。
- shape='valid'：不考虑边界补零，只计算有效数据。

4. filter2

MATLAB 图像处理工具箱提供了基于卷积的图像滤波函数 filter2，filter2 的语法格式为：

```
B=filter2(h,A)
B=filter2(h,A,shape)
```

其中 B=filter2(h,A)返回图像 A 经算子 h 滤波后的结果，参数 shape 指定滤波的计算范围，即：

- shape='full'时，作边界补零。
- shape='same'时，返回图像 B 与输入图像 A 大小相同。
- shape='valid'时，不考虑边界补零，只计算有效输出部分。

例如：

```
A=magic(6)
A=
    35     1     6    26    19    24
     3    32     7    21    23    25
    31     9     2    22    27    20
     8    28    33    17    10    15
    30     5    34    12    14    16
```

```
     4    36    29    13    18    11
h=fspecial('sobel')
h=
     1     2     1
     0     0     0
    -1    -2    -1
B=filter2(h,A,'valid')
B=
    -8     4     4    -8
   -23   -44    -5    40
   -23   -50     1    40
    -8     4     4    -8
```

5. fspecial

fspecial 函数用于创建预定义的滤波算子，其语法格式为：

```
h=fspecial(type)
h=fspecila(type,para)
```

参数 type 指定算子的类型，para 指定相应的参数，具体意义如下：

- type='average'，为均值滤波器，参数为 n，代表模板尺寸，用向量表示，默认值为[3 3]。
- type='gaussian'，为高斯低通滤波器，参数有两个， n 表示模板尺寸，默认值为[3 3]，sigma 为滤波器的标准差，单位为像素，默认值为 0.5。
- type='laplacian'，为拉普拉斯算子，参数为 alpha，用于控制拉普拉斯算子的形状，取值范围为[0,1]，默认值为 0.2。
- type='log'，为拉普拉斯高斯算子，参数有两个，n 表示模板尺寸，默认值为[3 3]，sigma 为滤波器的标准差，单位为像素，默认值为 0.5。
- type='prewit'，为 Prewitt 算子，用于边缘增强，无参数。
- type='sobel'，为著名的 Sobel 算子，用于边缘提取，无参数。
- type='unsharp'，为对比度增强滤波器，参数 alpha 用于控制滤波器形状，范围为[0,1]，默认值为 0.2。

7.4 平滑滤波

平滑技术用于平滑图像中的噪声。平滑噪声可以在空间域中进行，基本方法是求像素灰度的平均值或中值。为了既平滑噪声又保护图像信号，也有一些改进的技术，比如在频域中运用低通滤波技术。

实际获得的图像一般都因受到某种干扰而含有噪声。引起噪声的原因有敏感元器件的内部噪声、照相底片上感光材料的颗粒、传输通道的干扰及量化噪声等。噪声产生的原因决定了噪声的分布特性及它和图像信号的关系。

根据噪声和信号的关系可以将其分为两种形式：

- 加性噪声：有的噪声与图像信号 g(x,y)无关，在这种情况下，含噪图像 f(x,y)可表

示为：

```
f(x,y)=g(x,y)+n(x, y)
```

信道噪声及扫描图像时产生的噪声都属加性噪声。

- 乘性噪声：有的噪声与图像信号有关。这可以分为两种情况：一种是某像素处的噪声只与该像素的图像信号有关，另一种是某像点处的噪声与该像点及其邻域的图像信号有关。例如用飞点扫描器扫描图像时产生的噪声就和图像信号相关。如果噪声和信号成正比，则含噪图像 f(x,y)可以表示为：

```
f(x,y)=g(x,y)+n(x,y)g(x,y)
=(1+n(x,y))g(x,y)=n1(x,y)g(x,y)
```

另外，还可以根据噪声服从的分布对其进行分类，这时可以分为高斯噪声、泊松噪声和颗粒噪声等。泊松分布噪声一般出现在照度非常小及用高倍电子线路放大的情况下，椒盐噪声可以认为是泊松的噪声。其他的情况通常为加性高斯噪声。颗粒噪声可以认为是一白噪声过程，在密度域中是高斯分布加性噪声，而在强度域中为乘性噪声。

MATLAB 图像处理工具箱提供了模拟噪声生成的函数 imnoise，它可以对图像添加一些典型的噪声。

1. imnoise

imnoise 的语法格式为：

```
J=imnoise(I,type)
J=imnoise(I,type,parameters)
```

其中 J=imnoise(I,type)返回对原始图像 I 添加典型噪声的有噪图像 J。

参数 type 和 parameters 用于确定噪声的类型和相应的参数。

下面的命令是对图像 eight.tif 分别加入高斯噪声、椒盐噪声和乘性噪声，其结果如图 7.10 所示。

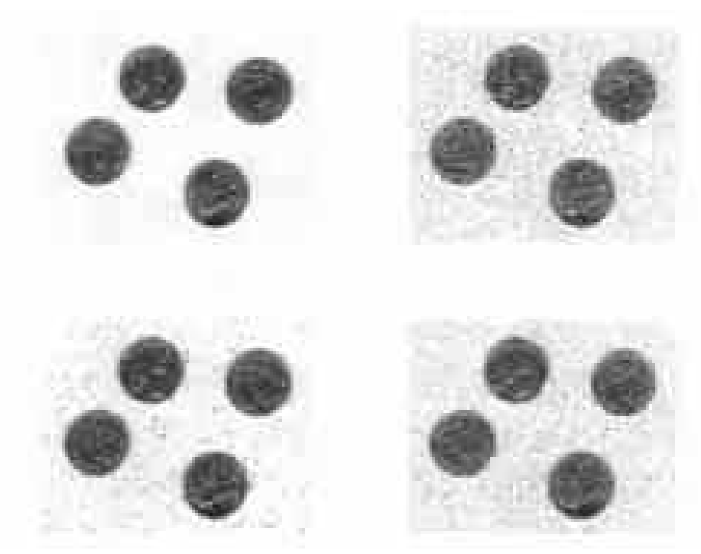


图 7.10 加入高斯噪声、椒盐噪声和乘性噪声的图像

```
I=imread('eight.tif');
```

```
J1=imnoise(I,'gaussian',0,0.02);
J2=imnoise(I,'salt&pepper',0.02)
J3=imnoise(I,'speckle',0.02);
subplot(2,2,1),imshow(I)
subplot(2,2,2),imshow(J1)
subplot(2,2,3),imshow(J2)
subplot(2,2,4),imshow(J3)
```

由于噪声的随机性，它们对某一像点的影响将使其灰度和邻点的灰度显著不同，因此可以利用这种不同来消除噪声。

MATLAB 图像处理工具箱提供了多种去除图像噪声的方法，共有下列 3 种：

- 线性滤波
- 中值滤波
- 自适应滤波

7.4.1 线性滤波

对一些图像进行线性滤波可以去除图像中某些类型的噪声，如采用领域平均法的均值滤波器就非常适用于去除通过扫描得到的图像中的颗粒噪声。

邻域平均法是空间域平滑噪声技术。对于给定的图像 f(i,j)中的每个像点(m,n)，取其邻域 S。设 S 含有 M 个像素，取其平均值作为处理后所得图像像点(m,n)处的灰度。用一像素邻域内各像素灰度平均值来代替该像素原来的灰度，即是邻域平均技术。

邻域 S 的形状和大小根据图像特点确定。一般取的形状是正方形、矩形及十字形等，S 的形状和大小可以在全图处理过程中保持不变，也可根据图像的局部统计特性而变化，点(m,n)一般位于 S 的中心。如 S 为 3×3 邻域，点(m,n)位于 S 中心，则：

$$\bar{f}(m,n)=\frac{1}{9}\sum_{i=-1}^{1}\sum_{j=-1}^{1}f(m+i,n+j)$$

假设噪声 n 是加性噪声，在空间各点互不相关，且期望为 0，方差为σ^2，g 是未受污染的图像，含有噪声的图像 f 经过邻域平均后为

$$\bar{f}(m,n)=\frac{1}{M}\sum f(i,j)=\frac{1}{M}\sum g(i,j)+\frac{1}{M}\sum n(i,j)$$

由上式可知，经邻域平均后，噪声的均值不变，方差$\sigma_a^2=\frac{1}{M}\sigma^2$，即噪声方差变小，说明噪声强度减弱了，即抑制了噪声。

由上式还可看出，邻域平均法也平滑了图像信号，特别是可能使图像目标区域的边界变得模糊。可以证明，对图像进行邻域平均处理相当于图像信号通过一低通滤波器。

下面的命令是对一幅含噪图像进行去噪处理，其结果如图 7.11 所示。

```
I=imread('eight.tif');
I=imnoise(I,'gaussian',0,0.02);
%添加均值为 0，方差为 0.02 的高斯噪声。
Imshow(I);
```

```
h=[ 1 1 1
    1 1 1
    1 1 1];
%产生滤波模板。
h=h/9;
%对滤波模板归一化。
J=conv2(I,h);
%用均值模板对图像滤波。
figure,imshow(J,[])
```

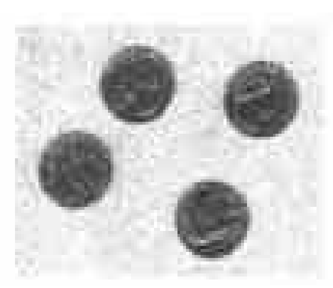
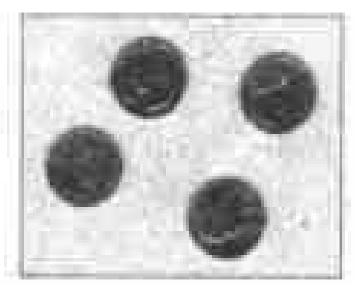



图 7.11　对噪声图像进行均值滤波的结果

7.4.2　中值滤波

中值滤波是抑制噪声的非线性处理方法。对于给定的 n 个数值{a1,a2,...an}，将它们按大小有序排列。当 n 为奇数时，位于中间位置的那个数值称为这 n 个数值中值。当 n 为偶数时，位于中间位置的两个数值的平均值称为这 n 个数值的中值，记作 med(a1，a2，…an)。中值滤波就是这样的一个变换，图像中滤波后某像素的输出等于该像素邻域中各像素灰度的中值。

中值滤波的方法运算简单，易于实现，而且能较好地保护边界，但有时会失掉图像中的细线和小块的目标区域。

邻域的大小决定在多少个数值中求中值，窗口的形状决定在什么样的几何空间中取元素计算中值。对二维图像，窗口 A 的形状可以是矩形、圆形及十字形等，它的中心一般位于被处理点上。窗口大小及形状有时对滤波效果影响很大。

一维信号中值滤波具有如下重要性质：

- 输入是阶跃信号或斜坡信号时，输出信号和输入信号相同。
- 若输入是脉宽小于窗口一半的脉冲 p，则该脉冲被滤除，否则输出和输入相同。
- 输入是三角形信号时，输出时其顶部被削平。

二维信号的中值滤波的性质与之类似。

在 MATLAB 图像处理工具箱中，提供了 medfilt2 函数用于实现中值滤波。

1.　medfilt2

medfilt2 函数的语法格式为：

```
B=medfilt2(A)                   %用于 3×3 的滤波窗口对图像 A 进行中值滤波。
B=medfilt2(A,[m,n])             %用指定大小为 m×n 的窗口对图像 A 进行中值滤波。
B=medfilt2(A,'indexed',…)       %对索引色图像 A 进行中值滤波。
```

图像 A 的数据类型可以是 double 型，也可以是 uint8 型。

例如对加入椒盐噪声的图像 eight.tif 作中值滤波，结果如图 7.12 所示。

```
I=imread('eight.tif');
J=imnoise(I,'salt&pepper',0.02);
K=medfilt2(J);
Subplot(1,2,1),imshow(J)
Subplot(1,2,2),imshow(K)
```

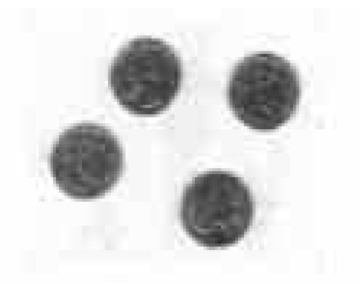


图 7.12 噪声图像中值滤波结果

从图中可以看出，中值滤波对于滤除图像的椒盐噪声非常有效，在去噪图像上椒盐噪声的斑点全部被去除。

另外 MATLAB 图像处理工具箱还提供了二维统计顺序滤波函数 ordfilt2。二维统计顺序滤波是中值滤波的推广，对于给定的 n 个数值{a1,a2,...an}，将它们按大小顺序排列，将处于第 k 个位置的元素作为图像滤波输出，即序号为 k 的二维统计滤波。

2. ordfilt2

ordfilt2 函数的语法格式为：

```
Y=ordfilt2(X,order,domain)
Y=ordfilt2(X,order,domain,S)
```

这里 Y=ordfilt2(X,order,domain)对图像 X 作顺序统计滤波，order 为滤波器输出的顺序值，domain 为滤波窗口。

S 是与 domain 大小相同的矩阵，它是对应 domain 中非零值位置的输出偏置，这在图像形态学中很有用。

例如： Y= ordfilt2(X,5,ones(3,3))，相当于 3×3 的中值滤波。

Y=ordfilt2(X,1,ones(3,3))，相当于 3×3 的最小值滤波。

Y=ordfilt2(X,9,ones(3,3))，相当于 3×3 的最大值滤波。

Y=ordfilt2(X,1,[0 1 0;1 0 1;0 1 0])的输出是每个像素的东、西、南、北 4 个方向相邻像素灰度的最小值。

7.4.3 自适应滤波

MATLAB 图像处理工具箱中的 wiener2 函数可以实现图像噪声的自适应滤除。wiener2 函数根据图像的局部方差来调整滤波器的输出，当局部方差大时，滤波器的平滑效果较小，滤波器平滑效果强。

wiener2 函数提供的自适应滤波通常比线性滤波的效果好，它比相应的线性滤波器具有更好的选择性，可以更好地保存图像的边缘和高频细节信息。另外，使用起来非常方便，wiener2 函数同时计算出滤波器的参数，并对图像进行滤波计算，而且 wiener2 函数并不比线性滤波器需要更多的计算时间。

wiener2 函数通常对于含有白色噪声(white noise)的图像滤波效果较好，比如含有高斯白噪声的图像。

wiener2 函数采用的算法是首先估计出象素的局部矩阵和方差：

$$\mu = \frac{1}{MN}\sum_{n_1,n_2\in\eta} a(n_1,n_2)$$

$$\sigma^2 = \frac{1}{MN}\sum_{n_1,n_2\in\eta} a^2(n_1,n_2) - \mu^2$$

η 是图像中每个象素的 M×N 的邻域。然后，对每一个象素利用 wiener 滤波器估计出其灰度值：

$$b(n_1,n_2) = \mu + \frac{\sigma^2 - \nu^2}{\sigma^2}(a(n_1,n_2) - \mu)$$

这里 ν^2 是图像中噪声的方差。

wiener2 的语法格式为：

```
J=wiener2(I,[m,n])
J=wiener2(I,[m,n],noise)
[J,noise]=wiener2(I,[m,n])
```

J=wiener2(I,[m,n])返回有噪图像 I 经过 wiener(维纳)滤波后的图像，[m,n]指定滤波器的窗口大小为 m×n，默认值为 3×3。

J=wiener2(I,[m,n],noise)指定噪声的功率，[J,noise]=wiener2(I,[m,n])在图像滤波的同时，返回噪声功率的估计值 noise。

例如对加入高斯噪声的图像 saturn.tif 作维纳滤波，结果如图 7.13 所示。

```
I=imread('saturn.tif')
J=imnoise(I,'gaussian',0,0.005)
K=wiener2(J,[5 5])
subplot(1,2,1), imshow(J)
subplot(1,2,2), imshow(K)
```

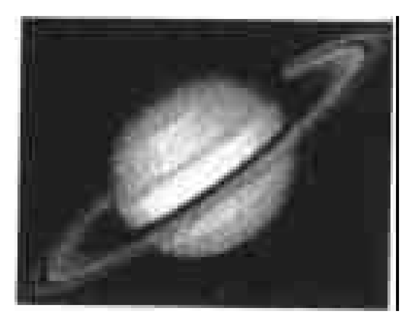
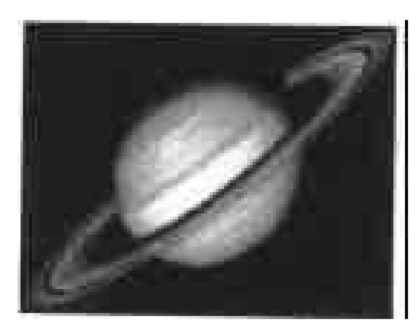



图 7.13　噪声图像维纳滤波前后结果

7.5 锐 化

锐化技术用于加强图像中的目标边界和图像细节。锐化技术可以在空间域中进行，常用的方法是对图像进行微分处理，也可以在频域中运用高通滤波技术。

7.5.1 模糊机理及解决方法

图像模糊是常见的图像降质问题。在图像摄取、传输及处理过程中有许多因素可以使图像变模糊。如光的衍射、聚焦不良、景物和取像装置的相对运动都会使图像变模糊，电子系统高频性能不好也会损失图像高频分量，而使图像不清晰。在对图像进行数字化时，实际取样点总是有一定的面积，所得的样本是这个具有一定面积的区域的亮度平均值，若取样点正好在边界上，则使样本值降低，从而使数字图像的边界变得不清楚。

大量的研究表明，各种图像变模糊的物理过程的数学模型一般含有求和、平均或积分运算。在某些应用中，可以不必深究图像变模糊的物理过程及其数学模型，而根据各种图像变模糊的过程都有相加或积分运算这一共同点，在空间域中运用微分运算增强图像，由付氏变换性质 $F\left[\frac{\partial}{\partial x}f(x,y)\right]=j2\pi uF[(x,y)]$ 可知，也可在频域中用加强信号高频分量的方法增强图像。

7.5.2 梯度模算子

因为需要锐化的边界可能是任意走向的，因此需要锐化算子是无方向的，即无论边界是什么走向，只要幅度相同，算子的输出就相同。这就可以证明梯度模算子和拉氏算子都是无方向性的。

由数学分析可知，可微函数 f(x,y)的方向导数 $\frac{\partial f}{\partial \alpha}$ 表示其在某一方向 α 上的变化率，且：

$$\frac{\partial f}{\partial \alpha}=\frac{\partial f}{\partial x}\cos\alpha+\frac{\partial f}{\partial y}\sin\alpha=G\bullet(\cos\alpha i+\sin\alpha j)$$

这里，

$$G\xrightarrow{\Delta}\frac{\partial f}{\partial x}i+\frac{\partial f}{\partial y}j$$

称为 f(x,y)的梯度，显然当 $\alpha=\tan^{-1}\left[\frac{\partial f}{\partial y}/\frac{\partial f}{\partial x}\right]$ 时，方向导数取最大值，这个最大值等于梯度的模：

$$|G|=\left[\left(\frac{\partial f}{\partial x}\right)^2+\left(\frac{\partial f}{\partial y}\right)^2\right]^{\frac{1}{2}}$$

对一幅图像施加梯度模算子，可以增强灰度变化的幅度，因此我们可以采用梯度模算子作为图像的锐化算子。而且梯度模算子具有方向同性和位移不变性。

对于离散函数 f(i,j)，也可以采用相似的概念，只是利用差分来代替微分。在 x 和 y 的一阶差分的定义为：

$$\Delta_x f(i,j)=f(i,j)-f(i-1,j)$$

$$\Delta_y f(i,j)=f(i,j)-f(i,j-1)$$

因此梯度模的定义为：

$$|G|=[\Delta_x f(i,j)^2+\Delta_y f(i,j)^2]^{\frac{1}{2}}$$

为了运算简便，实际中采用梯度模的近似形式，如：$|\Delta_x f(i,j)|+|\Delta_y f(i,j)|$、$\max(|\Delta_x f(i,j)|,|\Delta_y f(i,j)|)$ 、 $\max|f(i,j)-f(m,n)|$ 等。另外，还有一些常用的算子，如 Roberts 算子和 Sobel 算子。

Roberts 算子的表达式为：

$$\max(|f(i,j)-f(i+1,j+1)|,|f(i+1,j)-f(i,j+1)|)$$

Sobel 算子的表达式为：

$$\begin{pmatrix}1 & 2 & 1\\ 0 & 0 & 0\\ -1 & -2 & -1\end{pmatrix} \qquad \begin{pmatrix}1 & 0 & -1\\ 2 & 0 & -2\\ 1 & 0 & -1\end{pmatrix}$$

x 方向算子　　　　y 方向算子

Sobel 梯度为：

$$G(f(i,j))=(G_x(f(i,j))^2+G_y(f(i,j))^2)^{\frac{1}{2}}$$

Sobel 算子的特点是对称的一阶差分，对中心加权，具有一定的平滑作用。利用 Sobel 算子对图像滤波的例子如下，最后的结果如图 7.14 所示。

```
load imdemos flower
I=flower;
h=fspecial('sobel');
Imshow(I);
J=filter2(h,I);
figure,imshow(J,[])
```

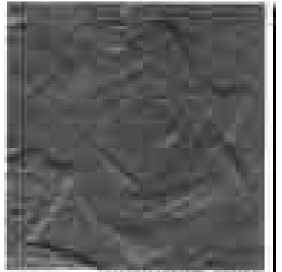



图 7.14 Sobel 算子对图像锐化结果

7.5.3 拉氏算子

拉氏算子比较适用于改善因为光线的漫反射造成的图像模糊。其原理是这样的，在摄影胶片记录图像的光化过程中，光点将光漫反射到其周围区域，这个过程满足扩散方程：

$$\frac{\partial f}{\partial t}=k\nabla^2 f$$

经过推导，可以发现当图像的模糊是由光的漫反射造成时，不模糊图像等于模糊图像减去它的拉氏变换的常数倍。另外，人们还发现，即使模糊不是由于光的漫反射造成的，对图像进行拉氏变换也可以使图像更清晰。

拉氏算子的表达式为：

$$\nabla^2 f=\frac{\partial^2 f}{\partial x^2}+\frac{\partial^2 f}{\partial y^2}$$

对于离散函数 f(i,j)，拉氏算子定义为：

$$\nabla^2 f(i,j)=\Delta_x^2 f(i,j)+\Delta_y^2 f(i,j)$$

这里 $\Delta_x^2 f(i,j)$ 和 $\Delta_y^2 f(i,j)$ 是 f(i,j)在 x 方向和 y 方向的二阶差分，所以离散函数的拉氏算子的表达式为：

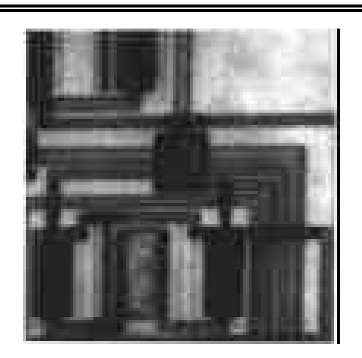
$$\nabla^2 f(i,j)=f(i+1,j)+f(i-1,j)+f(i,j+1)+f(i,j-1)-4f(i,j)$$

拉氏算子还可以用下面的模板来表示：

$$\begin{bmatrix}0 & 1 & 0\\ 1 & -4 & 1\\ 0 & 1 & 0\end{bmatrix}$$

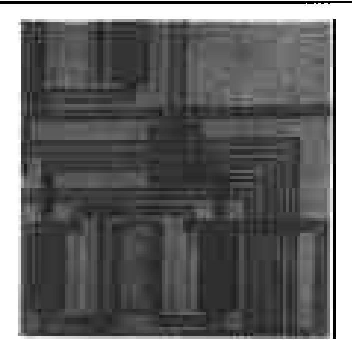
利用拉氏算子对模糊图像进行增强的例子如下，结果如图 7.15 所示。

```
load imdemos circuit
%读入图像文件。
I=circuit;
I=double(I);
%卷积运算不支持 uint8 类型。
%所以将图像矩阵转化为 double 类型。
imshow(I,[])
h=[0 1 0
   1 -4 0
   0 1 0];
J=conv2(I,h,'same');
%用拉氏算子对图像滤波。
K=I-J;
%增强图像为原始图像减去拉氏算子滤波的图像。
imshow(K,[])
```





原始模糊图像



经拉氏算子增强的图像

图 7.15　原始图像及经过增强的图像

比较原始模糊图像和经拉氏算子运算的图像，可以发现，图像模糊的部分得到了锐化，特别是模糊的边缘部分得到了增强，边界更加明显。但是，图像显示清楚的地方，经滤波后发生了失真，这也是拉氏算子增强的一大缺点。

7.6　光照不均的校正

假设图像是由光的反射形成的，如果光源照射到景物上的照度不均，那么照度较强的部分将较亮，照度较弱的部分就较暗，并且由此引起较暗部分的图像细节不易看清。

通常，对光照不均图像的校正要采用同态滤波的方法。我们知道，由光的反射形成的图像的数学模型为：

$$f(x,y)=r(x,y)i(x,y)$$

一般照度分量 i(x,y)是均匀的或者缓变的，其频谱分量落在低频区域，反射分量 r(x,y)反映图像的细节内容，它的频谱有较大的部分落在高频区域。同态滤波就是对图像取对数运算，将乘积模型转化为加性模型。经过分析，取对数运算后，照度分量和反射分量所处区域不变，从而对数域将照度分量和反射分量区分开来。这时就可以根据需要对照度分量和反射分量进行调整，通常为了消除照度不均的影响，应衰减照度分量的频率成份，另一方面，为了更清楚地显示景物暗区的细节，应该对反射分量进行增强。

对于一般照度不均的图像，还可以采用下面简单的方法来消除其影响，下面举例说明。

有这样一幅图像，如图 7.16 所示，图像的下部的灰度比上部和中部要低。

```
rice=imread('rice.tif')
rice=im2double(rice)
imshow(rice)
```

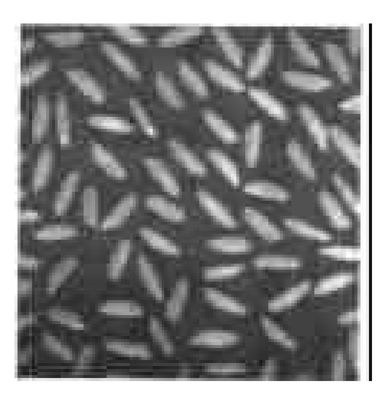


图 7.16　原始照度不均图像

首先，如图 7.17 所示，估计出图像背景的照度。方法是取 32×32 大小的图像块中的最小值作为图像背景的照度。这里利用来 blkproc 函数来加快运算速度。

```
bg32=blkproc(rice,[32 32],'min(x(:))')
```

```
surf(bg32)
```

然后将粗略估计出的背景照度矩阵扩展成和原始图像大小相同的矩阵，这可以通过双三次插值实现，结果如图 7.18 所示。

```
bg256=imresize(bg32,[256 256],'bicubic')
imshow(bg256)
```

将估计出的背景照度从原始图像中减去，即可修正照度不均的影响，但是这样作的后果是图像变暗，如图 7.19 所示。

```
d=rice-bg256
imshow(d)
```

这种后果可以通过调整图像的灰度来进行校正。我们可以指定图像的灰度范围，然后调用 imadjust 函数进行调整。调整之后的图像，米粒变亮，而且可以看到更多的细节，如图 7.20 所示。

```
adjusted=imadjust(d,[0 max(d(:))],[0 1],1)
imshow(adjusted)
```

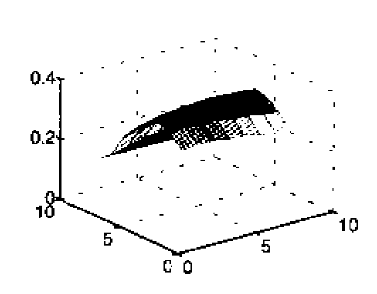


图 7.17 背景灰度估计曲面

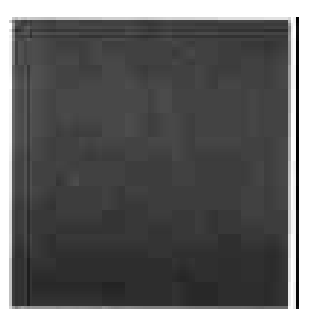


图 7.18 背景照度矩阵扩展结果

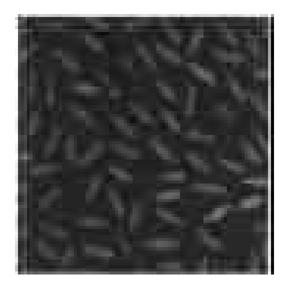


图 7.19 从原始图像中减去背景照度的结果

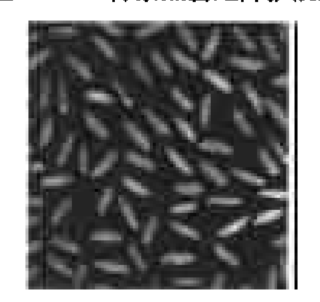


图 7.20 图像对比度调整结果

7.7 利用小波分析工具箱去除图像噪声

7.7.1 小波去噪原理

图像降噪方法有时域和频域两种，其工件原理是利用噪声和信号在频域上分布的不同进行的。信号主要分布在低频区域，而噪声主要分布在高频区域，但同时图像的细节也分布在高频区域。在传统的基于傅氏变换的信号去噪方法中，我们使得信号和噪声的频带重

叠部分尽可能较小，这样就可以在频域通过时不改变滤波，就将信号同噪声区分开。但是当它们的频域重叠区域很大时，这种方法就无能为力了。所以，图像降噪处理中一个矛盾的问题是如何在降低图像噪声和保留图像细节上保持平衡，传统的低通滤波方法将图像的高频成分滤除，虽然能够达到降低噪声的效果，但破坏了图像细节。利用小波分析的理论，可以构造一种既能够降低图像噪声，又能够保持图像细节信息的方法。

假设已经获得信号的观测公式如下：

$$y_i = x_i + n_i, i = 1,2,...M$$

其中 n_i 为零均值的白色高斯噪声，σ 为其方差，x_i 为期望信号，y_i 为观测值。滤除噪声 n_i 的问题可以认为是如何将 x 从观测值 y 中恢复。

假设离散小波变换的变换矩阵为 W，则对上进行小波变换得到：

$$Y = X + N$$

这里，$Y = W[y_i]$，$X = W[x_i]$，$N = W[n_i]$。对应于 W，存在逆变换矩阵 M，满足 WM =I。

由小波变换的特性可知，高斯噪声的小波变换仍然是高斯分布的，它均匀分布在频率尺度空间的各部分，而信号由于其带限性，它的小波变换系数仅仅集中在频率尺度空间上的有限部分，这样，从信号能量的观点来看，在小波域上，所有的小波系数都对噪声有贡献，也就是噪声的能量分布在所有的小波系数上，而只有一小部分小波系数对信号能量有贡献，所以可以把小波系数分成两类，第一类小波系数仅仅由噪声变换后得到，这类小波系数幅值小，数目较多。第二类小波系数由信号变换得来，并包含噪声的变换结果，这类小波系数幅值大，数目较小。根据信号小波分界的这个特点，可以通过这种小波系数幅值上的差异来降低噪声。对信号的小波系数，设置一个阈值，大于这个阈值的小波系数认为属于第二类系数，它同时含有信号和噪声的变换结果，可以简单保留或进行后续操作，而小于这个阈值的小波系数，则认为是第一类小波系数，即完全由噪声变换而来，应该去掉这些系数。这样达到了降低噪声的目的。同时由于这种方法保留大部分包含信号的小波系数，因此可以较好地保持图像细节。

7.7.2 MATLAB 提供的去噪和压缩函数

MATLAB 的小波分析工具箱提供了对信号进行去噪和压缩的一族函数。小波去噪和小波压缩的主要区别在于选择的阈值准则不同，在操作时可以采用全局阈值，也可以采用自适应阈值。

1. ddencmp

ddencmp 函数用于自动生成小波去噪或者压缩的阈值选取方案：

```
[THR,SORH,KEEPAPP,CRIT]=ddencmp(IN1,IN2,X)
[THR,SORH,KEEPAPP]=ddencmp(IN1,'wv',X)
[THR,SORH,KEEPAPP,CRIT]=ddencmp(IN1,'wp',X)
```

[THR,SORH,KEEPAPP,CRIT]=ddencmp(IN1,IN2,X)根据信号 X 和指定的参数 IN1 和

IN2 自动生成利用小波分解去噪或者压缩的阈值选择方案。

IN1 决定使用的目的，即：

- IN1='den'：用于去除信号噪声。
- IN1='cmp'：用于压缩信号。

IN2 指定使用小波分解还是小波包分解的方法，参数含义如下：

- IN2='wv'：指定使用小波分解。
- IN2='wp'：指定使用小波包分解。

返回值 THR 是生成的小波去噪或者压缩的阈值。

SORH 决定阈值的使用方式，具体内容如下：

- SORH='s'：使用软阈值。
- SORH='h'：使用硬阈值。

KEEPAPP 指定是否对近似分量进行阈值处理，KEEPAPP=0 不进行阈值处理，KEEPAPP =1 进行阈值处理。

CRIT 为使用小波包分解时采用的熵函数的类型。

2. wden

wden 函数用于一维信号的小波去噪，其语法格式为：

```
[XD,CXD,LXD]=wden(X,TPTR,SORH,SCAL,N,'wname')
[XD,CXD,LXD]=wden(C,L,TPTR,SORH,SCAL,N,'wname')
```

[XD,CXD,LXD]=wden(X,TPTR,SORH,SCAL,N,'wname')对输入信号 X 进行去噪处理，返回经过处理的信号 XD，以及 XD 的小波分解结构[CXD,LXD]。

用户可以指定阈值选择算法，其中：

- TPTR='rigrsurc'时，选择基于 stein 无偏估计理论的自适应阈值。
- TPTR='hursure'时，选择第一种阈值选择方式的启发式改进形式。
- TPTR='sqtwolog'时，选择全局阈值 $\sqrt{2\log(\bullet)}$ 。
- TPTR='minimaxi'时，选用极小极大准则确定阈值。

SORH 决定阈值的使用方式。

- SORH='s'，使用软阈值。
- SORH='h'，使用硬阈值。

SCAL 决定阈值处理是否随噪声变化，其中：

- SCAL='one'，不随噪声方差变化。
- SCAL='sln'，阈值根据第一层小波分解的噪声方差调整。
- SCAL='mln'，根据各层小波分解的噪声方差调整阈值，这适合于色噪声的情况。

[XD,CXD,LXD]=wden(C,L,TPTR,SORH,SCAL,N,'wname')根据信号小波分解结构[C,L]对信号进行去噪处理。

3. wdencmp

wdencmp 函数用于信号的小波分解去噪或者压缩。

```
[XC,CXC,LXC,PERF0,PERFL2]
=wdencmp('gbl',X,'wname',N,THR,SORH, KEEPAPP)
[XC,CXC,LXC,PERF0,PERFL2]
= wdencmp('lvd',X,'wname',N,THR,SORH)
[XC,CXC,LXC,PERF0,PERFL2]
 = wdencmp('lvd',C,L,'wname',N,THR,SORH)
```

wdencmp 用于一维或者二维的基于小波分解的信号去噪或者压缩。

[XC,CXC,LXC,PERF0,PERFL2] =wdencmp('gbl',X,'wname',N,THR, SORH, KEEPAPP) 返回采用全局阈值对信号 X 进行去噪或者压缩后的信号 XC，以及它的小波分解结构 CXC 和 LXC。THR 是用于小波去噪或者压缩的阈值，SORH 决定阈值的使用方式，含义与 ddencmp 函数中的含义相同。

KEEPAPP 指定是否对近似分量进行阈值处理，KEEPAPP=0，不进行阈值处理；KEEPAPP=1，进行阈值处理。

返回值 PERF0,PERFL2 是用 $L-2$ 范数度量的信号的恢复率和压缩率。

[XC,CXC,LXC,PERF0,PERFL2] = wdencmp('lvd',C,L,'wname',N,THR, SORH)利用原信号的小波分解结构 C,L 对信号进行去噪和压缩。

4. wpdencmp

wpdencmp 函数用于基于小波包分解的信号去噪或者压缩。

```
wpdencmp
[XD,TREED,PERF0,PERFL2]
=wpdencmp(X,SORH,N,'wname',CRIT,PAR, KEEPAPP)
[XD,TREED,PERF0,PERFL2]
=wpdencmp(TREE,SORH,CRIT,PAR,KEEPAPP)
[XD,TREED,PERF0,PERFL2]
=wpdencmp(X,SORH,N,'wname',CRIT,PAR,KEEPAPP)
```

wpdencmp 函数用于基于小波包分解的信号去噪或者压缩。函数返回值 XD 为处理后的信号，[TREED,DATA]为原信号的小波包分解树结构，PERF0, PERFL2 为采用 $L-2$ 范数度量的信号的恢复率和压缩率。

'wname'为采用的小波基函数，CRIT 和 PAR 为采用的熵函数及参数。

用户可以根据 SORH 选择阈值的实现方式，具体内容如下：

- SORH='s'，使用软阈值。
- SORH='h'，使用硬阈值。

5. wthresh

利用 wthresh 函数可以实现硬阈值和软阈值处理，其语法格式为：

```
Y=wthresh(X,SORH,T)
```

它根据由 SORH 指定的阈值处理方法对输入信号 X 进行阈值处理，T 为指定的阈值。

用户可以根据 SORH 选择阈值的实现方式：

- SORH='s'，使用软阈值。

- SORH='h'，使用硬阈值。

7.7.3 小波去噪和压缩的例子

MATLAB 中提供了一个产生有噪信号的函数，用于测试去噪算法。

1. wnoise

wnoise 函数的语法格式如下：

```
X=wnoise(NUM,N)
[X,XN]=wnoise(NUM,N,SNRAT)
[X,XN]=wnoise(NUM,N,SNRAT,INIT)
```

X=wnoise(NUM,N)产生幅值在[0,1]之间长度为 2^N 的信号，信号的类型由 NUM 指定：

- NUM=1，产生不规则方波信号。
- NUM=2，产生低频噪声。
- NUM=3，产生随机间断的正弦信号。
- NUM=4，产生 chirp 信号。
- NUM=5，产生 4 次调频信号。
- NUM=6，产生混杂信号。

[X,XN]=wnoise(NUM,N,SNRAT)产生含有白噪声的信号 XN，SNRAT 是信号的信噪比。

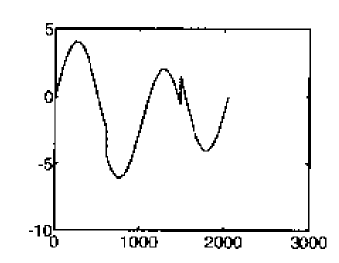
[X,XN]=wnoise(NUM,N,SNRAT,INIT)使用初始值 INIT 产生含噪信号。

(1) 首先产生一个长度为 2^{11} 点，包含高斯白噪声的正弦信号。噪声标准差为 3。

```
sqrt_snr=3;
init=231434;
[x,xn]=wnoise(3,11,sqrt_snr,init);
plot(x)
figure,plot(xn)
```

(2) 利用'sym8'小波对信号进行分解，在分解的第 5 层上，利用启发式 SURE 阈值选择算法，对信号进行去噪，如图 7.21 及图 7.22 所示。

```
lev=5;
xd=wden(x,'heursure','s','one',lev,'sym8');
plot(xd)
```



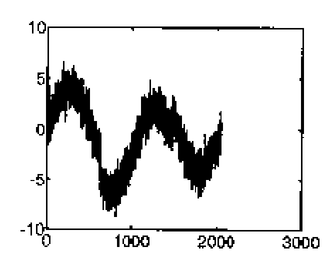


图 7.21 原始信号和含噪信号

(3) 在同样的条件下，用软 SURE 阈值选择算法对信号去噪，结果如图 7.23 所示。

```
xd=wden(x,'rigrsure','s','sln',lev,'sym8');
plot(xd)
```

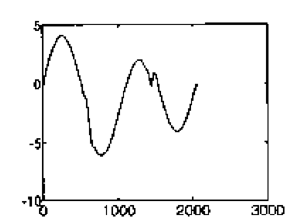


图 7.22　启发式SURE阈值去噪结果

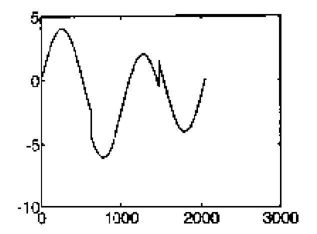


图 7.23　软SURE阈值去噪结果

(4) 同样的条件下，用固定阈值选择算法对信号去噪，结果如图 7.24 所示。

```
xd=wden(x,'sqtwolog','s','sln',lev,'sym8');
plot(xd)
```

(5) 利用信号小波分解的结构[C,L]对信号进行去噪处理。如果需要对信号进行多次去噪，这种方法比较方便，去噪后的结果如图 7.25 所示。

```
[c,l]=wavedec(x,lev,'sym8');
xd=wden(c,l,'minimaxi','s','sln',lev,'sym8');
plot(xd)
```

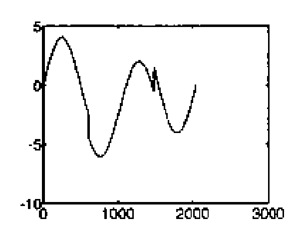


图 7.24　固定阈值去噪结果

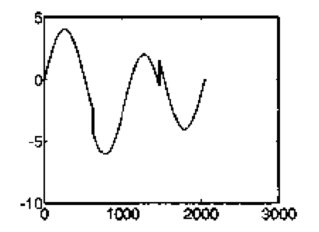


图 7.25　利用小波分解结构去噪结果

利用 MATLAB 提供的函数来对图像进行去噪处理也很方便，接下来用具体例子进行说明。

(1) 首先读入一幅图像，并给图像加入一定的噪声，原始图像及含噪图像分别如图 7.26 所示。

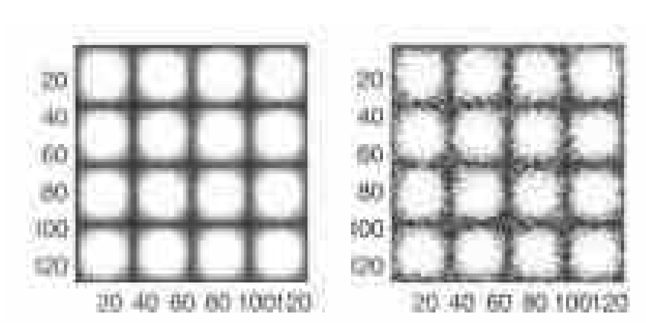


图 7.26　原始图像及含噪图像

```
load sinsin
colormap(pink(64))
image(X)
axis('square')
```

```
init=231434;
randn('seed',init);
x = X + 18*randn(size(X));
image(x)
axis('square')
```

(2) 为图像去噪选择方案，使用 sym4 小波，设定全局阈值对图像去噪处理，结果如图 7.27 所示。

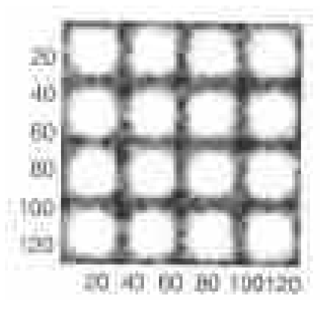


图 7.27 小波去噪结果

```
[thr,sorh,keepapp] = ddencmp('den','wv',x);
xd=wdencmp('gbl',x,'sym4',2,thr,sorh,keepapp);
image(xd)
axis('square')
```

从含噪图像可以看出噪声含量非常强，而从去噪的结果可以看出，利用小波分解去除图像噪声既滤除了噪声，又有效地保持了边界。

另外，MATLAB 小波分析工具箱还提供了更简单去除图像噪声的方法，即利用其图像用户界面(GUI)的工具对图像进行去噪和压缩操作。利用图像用户界面对图像处理的优点在于：

- 操作简便，不需记忆 MATLAB 提供的众多的专用函数。只要知道小波去噪的基本原理，就可以根据图像用户界面提供的各种功能对图像进行处理。
- 具有较好的交互功能，可以方便地改变去噪和压缩的参数，直观地观察效果，并不断改进，直到满意为止。

MATLAB 小波分析图像用户界面的使用步骤如下：

(1) 在 MATLAB 命令窗口里输入“wavemenu”，打开小波分析的图像用户界面窗口，如图 7.28 所示。



图 7.28 小波分析的图像用户界面窗口

(2) 单击 Wavelet2-D 按钮，选择 File 菜单里的命令读入图像，同时还可以选择分析用的小波基函数以及分解的层数，如图 7.29 所示。

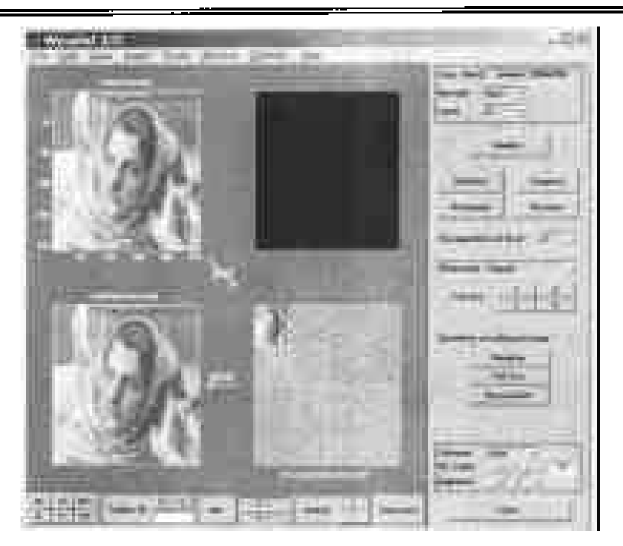


图 7.29　二维小波分析的图像用户界面窗口

(3) 单击 De-noise 按钮，在新产生的窗口中可以选择阈值处理的方法，还可以手动设定阈值，这样就可以一边修改阈值，一边观察去噪效果，直到去噪效果满意为止，如图 7.30 所示。

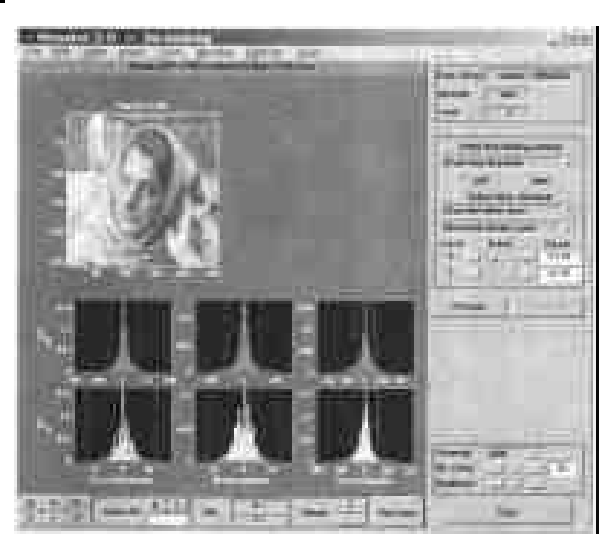


图 7.30　利用小波分析去除二维图像噪声窗口

第8章　边缘提取和图像分割

图像理解是图像处理的一个重要分支，它研究为完成某一任务需要从图像中提取哪些有用信息，以及如何利用这些信息解释图像。为了有效地分析和理解图像，或者对图像进行压缩，往往需要把给定的图像以及已经分割的图像区域用更为简单明确的数值、符号或图形表示出来。这些数值、符号或图形通常是根据一定的概念和公式从原图像中抽取出来的，提供了原图像的基本信息，反映了原图像的主要特性，可以使图像信息的表达更加简洁清晰。这些数值、符号或图形的选择和提取因为有利于人或机器对于原始图像的分析和理解，通常被称为图像的特征。产生这些特征的过程称为图像特征抽取，用这些特征表示图像称为图像描述。

在计算机视觉技术中，图像符号的描述主要包括图像的线特征描述和区域特征描述。本章重点讨论常用的边缘检测技术，特别是图像中直线提取的技术，以及图像分割技术，包括灰度门限法分割图像和利用四分树分解分割图像的技术。

8.1 边缘检测

边缘检测技术对于处理数字图像非常重要，因为边缘是所要提取目标和背景的分界线，提取出边缘才能将目标和背景区分开来。在图像中，边界表明一个特征区域的终结和另一个特征区域的开始，边界所分开区域的内部特征或属性是一致的，而不同区域内部的特征或属性是不同的，边缘的检测正是利用物体和背景在某种图像特性上的差异来实现的，这些差异包括灰度、颜色或者纹理特征。边缘检测实际上就是检测图像特性发生变化的位置。

由于噪声和模糊的存在，检测到的边界可能会变宽或在某些点处发生间断，因此，边界检测包括两个基本内容：首先抽取出反映灰度变化的边缘点，然后剔除某些边界点或填补边界间断点，并将这些边缘连接成完整的线。

MATLAB 图像处理工具箱提供的 edge 函数可以实现检测边缘的功能，其语法格式如下：

```
BW=edge(I,'sobel')
BW=edge(I,'sobel',thresh)
BW=edge(I,'sobel',thresh,direction)
[BW,thresh]=edge(I,'sobel',…)
BW=edge(I,'prewitt')
BW=edge(I,'prewitt',thresh)
BW=edge(I,'prewitt',thresh,direction)
[BW,thresh]=edge(I,'prewitt',…)
BW=edge(I,'Roberts算子')
```

```
BW=edge(I,'Roberts 算子',thresh)
[BW,thresh]=edge(I,'Roberts 算子',…)
BW=edge(I,'log')
BW=edge(I,'log',thresh)
BW=edge(I,'log',thresh,sigma)
[BW,thresh]=edge(I,'log',…)
BW=edge(I,'zerocross',thresh,h)
[BW,thresh]=edge(I,'zerocross',…)
BW=edge(I,'canny')
BW=edge(I,'canny',thresh)
BW=edge(I,'canny',thresh,sigma)
[BW,thresh]=edge(I,'canny',…)
```

这里 BW=edge(I,'sobel')采用 Sobel 算子进行边缘检测。

BW=edge(I,'sobel',thresh)指定阈值 thresh，默认时函数会利用 RMS 算法自动选取。

[BW,thresh]=edge(I,'sobel',…)根据默认的阈值进行边缘检测，并由 thresh 返回函数自动选取的门限值。用户可以在观察边缘检测效果的同时，根据返回的门限进行调整，直到满意为止。

BW=edge(I,'sobel',thresh,direction)还可以指定算子的方向，即：

- direction='horiznotal'，为水平方向。
- direction='vertical'，为垂直方向。
- direction='both'，为水平和垂直两个方向。

采用 Prewitt 算子进行边缘检测，各种调用方法与 Sobel 算子的方法类似，分别为 BW=edge(I,'prewitt')、BW=edge(I,'prewitt',thresh)、BW=edge(I,'prewitt', thresh,direction)和[BW,thresh]=edge(I,'prewitt',…)。

BW=edge (I,'Roberts 算子')、BW=edge (I,'Roberts 算子',thresh)和[BW,thresh]=edge (I,'Roberts 算子',…)采用 Roberts 算子进行边缘检测，各种调用方法与 Sobel 算子类似。

BW=edge(I,'log')、BW=edge(I,'log',thresh)、BW=edge(I,'log',thresh,sigma)和[BW,thresh]=edge(I,'log',…)采用拉普拉斯高斯(log)算子进行边缘检测，其中 sigma 为高斯滤波器的标准差，默认值为 2，其他各种调用方法与 Sobel 算子类似。

BW=edge(I,'zerocross',thresh,h)和[BW,thresh]=edge (I,'zerocross',…)采用过零点方法进行边缘检测，各种调用方法与 Sobel 算子类似。

BW=edge(I,'canny')、BW=edge(I,'canny',thresh)、BW=edge(I,'canny',thresh,sigma)和[BW,thresh]=edge(I,'canny',…)采用 Canny 算子进行边缘检测，其中 thresh 是双阈值向量，这两个元素分别指定大和小阈值。sigma 为所使用的高斯滤波器的标准差，默认值为 1。

下面对几种常用的算子的理论进行简单介绍。

8.1.1　微分算子法

导数算子具有突出灰度变化的作用，对图像运用导数算子，灰度变化较大的点处算得的值较高，因此可将这些导数值作为相应点的边界强度，通过设置门限的方法，提取边界点集。

一阶导数$\frac{\partial f}{\partial x}$与$\frac{\partial f}{\partial y}$是最简单的导数算子，它们分别求出了灰度在x和y方向上的变化率，而方向α上的灰度变化率可以用下面式子计算：

$$\frac{\partial f}{\partial \alpha}=\frac{\partial f}{\partial x}\cos\alpha+\frac{\partial f}{\partial y}\sin\alpha=G\bullet(\cos\alpha i+\sin\alpha j)$$

对于数字图像，应该采用差分运算代替求导，相对应的一阶差分为：

$$\Delta_x f(i,j)=f(i,j)-f(i-1,j)$$

$$\Delta_y f(i,j)=f(i,j)-f(i,j-1)$$

方向差分为：

$$\Delta_\alpha f(i,j)=\Delta_x f(i,j)\cos\alpha+\Delta_y f(i,j)\sin\alpha$$

函数 f 在某点的方向导数取得最大值的方向是$\alpha=\tan^{-1}\left[\frac{\partial f}{\partial y}/\frac{\partial f}{\partial x}\right]$，方向导数的最大值是$|G|=\left[\left(\frac{\partial f}{\partial x}\right)^2+\left(\frac{\partial f}{\partial y}\right)^2\right]^{\frac{1}{2}}$称为梯度模。利用梯度模算子来检测边缘是一种很好的方法，它不仅具有位移不变性，还具有各向同性。而由$\alpha=\tan^{-1}\left[\frac{\partial f}{\partial y}/\frac{\partial f}{\partial x}\right]$可以计算出灰度变化的方向，即边界的方向。边界的方向可以用于 LoG 算子和 Canny 算子提取边缘，也是后面讨论的相位编组法提取直线的关键。

为了运算简便，实际中采用梯度模的近似形式，如：$|\Delta_x f(i,j)|+|\Delta_y f(i,j)|$、$\max(|\Delta_x f(i,j)|,|\Delta_y f(i,j)|)$及$\max|f(i,j)-f(m,n)|$等。另外，还有一些常用的算子，如 Roberts 算子和 Sobel 算子。

Roberts 算子的表达式为：

$$\max(|f(i,j)-f(i+1,j+1)|,|f(i+1,j)-f(i,j+1)|)$$

Sobel 算子的表达式为：

$$\begin{pmatrix}1&2&1\\0&0&0\\-1&-2&-1\end{pmatrix}\qquad\begin{pmatrix}1&0&-1\\2&0&-2\\1&0&-1\end{pmatrix}$$

x方向算子　　　　y方向算子

其中，由于 Sobel 算子是滤波算子的形式，用于提取边缘。我们可以利用快速卷积函数，简单有效，因此应用很广泛。

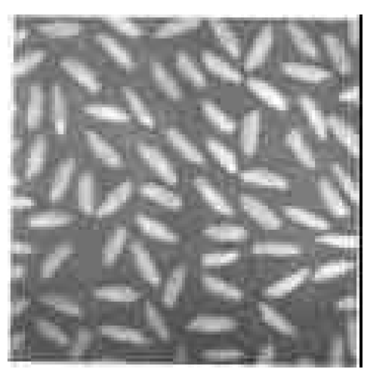
用 Roberts 算子和 Sobel 算子进行边缘提取的命令如下，其示意图如图 8.1 所示。

```
I=imread('rice.tif');
Imshow(I);
BW1=edge(I,'Roberts 算子');
figure, imshow(BW1)
BW=edge(I,'sobel');
figure,imshow(BW2)
```

由上图可以看出，利用 Roberts 算子提取边缘的结果边缘较粗，因此边缘定位不是很准

确，而 Sobel 算子对边缘的定位比较准确，这也是 Sobel 算子得到广泛应用的原因。

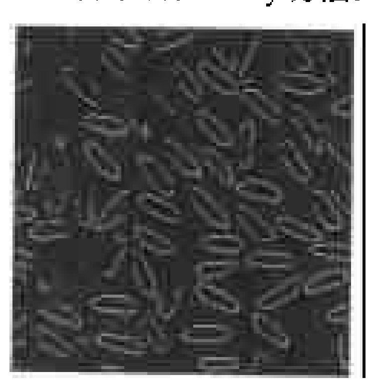
在实际图像中，对应景物边缘的图像灰度变化有时并不十分陡峭，另外，图像中也存在噪声，因此直接运用微分算子提取边界后，还需作某些处理(如连接及细化等)才能形成一条有意义的边界。下面就介绍两种改进的算法，即拉普拉斯高斯算子法和 canny 方法。



原始图像



Roberts算子边缘提取结果



Sobel算子边缘提取结果

图 8.1 Roberts算子和Sobel算子边缘检测结果

8.1.2 拉普拉斯高斯算子法

拉普拉斯高斯(LoG)算法是一种二阶边缘检测方法。它通过寻找图像灰度值中二阶微分中的过零点 (Zero Crossing)来检测边缘点。其原理为，灰度缓变形成的边缘经过微分算子形成一个单峰函数，峰值位置对应边缘点；对单峰函数进行微分，则峰值处的微分值为 0，峰值两侧符号相反，而原先的极值点对应二阶微分中的过零点，如图 8.2 所示，通过检测过零点即可将图像的边缘提取出来。

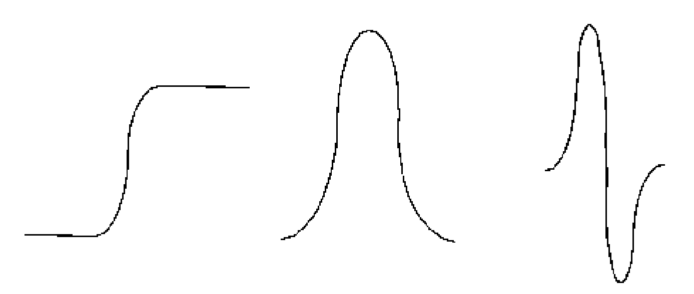


图 8.2 边缘、边缘的一阶微分和二阶微分

实际应用中，为了去除噪声影响，首先要用高斯函数对图像进行滤波，然后对滤波后的图像求二阶导数，即按照下式计算：

$$\nabla^2[G(x,y)*f(x,y)]$$

其中 f(x,y)为图像，G(x,y)为高斯函数，上述两个处理步骤可以合成一个算子，由卷积和微分可交换顺序的性质知：

$$\nabla^2[G(x,y)*f(x,y)]=\nabla^2G(x,y)*f(x,y)$$

式中 $\nabla^2G(x,y)$ 称为拉普拉斯高斯算子，经运算可得：

$$\nabla^2 G(x,y)=\frac{1}{2\pi\sigma^4}\left(\frac{x^2+y^2}{\sigma^2}-2\right)\exp\left\{-\frac{x^2+y^2}{2\sigma^2}\right\}$$

$\nabla^2 G(x,y)$是关于原点对称的函数，其主瓣宽度为：

$$W=2\sqrt{2}\sigma$$

利用 fspeical 函数可以产生拉普拉斯高斯算子，下面我们画出它的图形，LoG 算子的范围为−2～2，方差为 0.5，最后的效果如图 8.3 所示。

```
x=-2:0.05:2;
y=-2:0.05:2;
sigma=0.5;
y=y';
fo~ i=1:(4/0.05+1)
    xx(i,:)=x;
    yy(:,i)=y;
end
%产生矩形网格
r=1/(2*pi*sigma^4)*((xx.^2+yy.^2)/(sigma^2)-2).*...
    exp(-(xx.^2+yy.^2)/(2*sigma^2));
%计算 LoG 算子的值。
colormap(jet(16));
mesh(xx,yy,r)
%mesh 函数用于将函数值用三维网格显示。
```

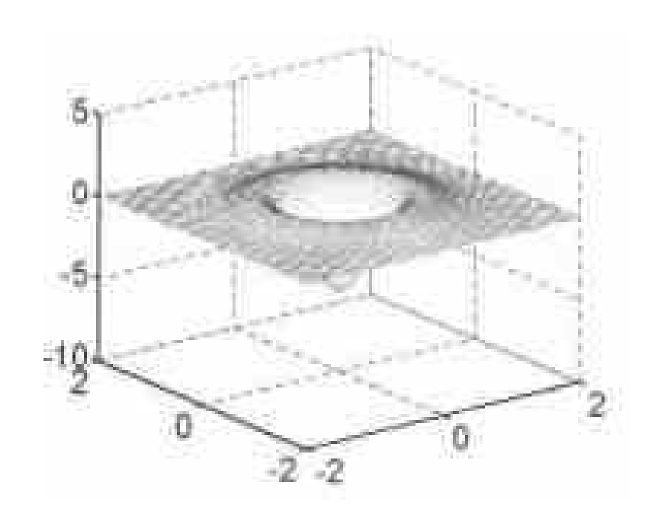


图 8.3 LoG算子可视化显示

由图 8.3 可以看出，它是一带通滤波器。研究表明，$\nabla^2 G(x,y)$比较符合人的视觉特性，这也说明了 LoG 算子较好地反映了视觉模型。

在实际应用中，可将$\nabla^2 G(x,y)$简化为

$$\nabla^2 G(x,y)=K\left(2-\frac{x^2+y^2}{\sigma^2}\right)\exp\left\{-\frac{x^2+y^2}{2\sigma^2}\right\}$$

在参数设计中，σ 取值较大时，趋于平滑图像；σ 较小时，则趋于锐化图像。通常应

根据图像特点通过实验选择合适的σ。$\nabla^2 G(x,y)$用 N×N 模板算子表示时，一般选择算子尺寸$N=(3\sim4)W$。K 的选取应使各阵元为整数且使所有阵元之和为零。

在这里，检测边界就是寻找$\nabla^2 G(x,y)$的过零点，可用以下几种参数表示过零点处灰度变化的速率：

- 过零点处的斜率。
- 二次微分峰——峰差值。
- 二次微分峰——峰间曲线下面积绝对值之和。

边界点方向信息可由梯度算子给出。为减小计算量，在实用中可用高斯差分算子(DOG)：

$$DOG(\sigma_1,\sigma_2)=\frac{1}{\sqrt{2\pi}\sigma_1}\exp\left\{-\frac{x^2+y^2}{2\sigma_1^2}\right\}-\frac{1}{\sqrt{2\pi}\sigma_2}\exp\left\{-\frac{x^2+y^2}{2\sigma_2^2}\right\}$$

代替$\nabla^2 G(x,y)$。

具体的边缘检测算法如下：

(1) 用拉普拉斯高斯滤波器对图像滤波，得到滤波图像。

(2) 对得到的图像进行过零检测，具体方法为：假定得到的图像的一阶微分图像的每个像素为 P[i, j]，L [i, j]为其拉普拉斯值。P 和 L 的含义如图 8.4 所示。

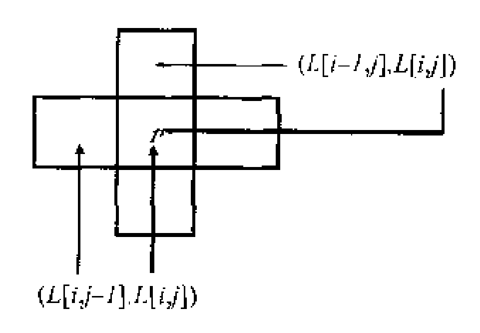


图 8.4　LoG算子过零检测示意图

接下来按照下面的规则进行判断：

- 如果 L [i, j]=0，则看数对([i- 1 , j], L [i+1 , j])或 (L [i, j- 1], L [i,j+1])中是否包含正负号相反的两个数。只要这两个数对中有一个包含正负号相反的两个数，则 P[i, j]是零穿越。然后看 P[i, j]对应得一阶差分值是否大于一定的阈值，如果是，则 P[i, j]是边缘点，否则不是。
- 如果 L [i, j]不为 0，则看 4 个数对 (L [i, j], L [i−1 , j](L [i, j],(L [i+1 , j]) , (L [i, j], L [i, j−1]) ,(L [i, j], L [i, j+1])中是否有包含正负号相反的值。如果有，那么在 P[i,j]附近有零穿越。看 P [i, j]对应的一阶差分值是否大于一定的阈值，如果是，则将 P[i, j]作为边缘点。

下面的代码可以实现 LoG 算子提取边缘点的功能：

```
function e=log_edge(a)
%该函数实现 LoG 算子提取边缘点，
%输入为图像 a，输出为边缘图像 e。
```

```
[m,n]=size(a);
e=repmat(logical(uint8(0)),m,n);
%产生同样大小的边缘图像 e，初始化为 0。
rr=2:m-1;cc=2:n-1;
fsize=ceil(sigma*3)*2+1;
% 选择点数为奇数的滤波器的尺寸 fsize>6*sigma;
op=fspecial('log',fsize,sigma);
%产生 LoG 滤波器。
op=op-sum(op(:))/prod(size(op));
%将 LoG 滤波器的均值变为 0。
b = filter2(op,a);
%利用 LoG 算子对图像滤波。
thresh=.75*mean2(abs(b(rr,cc)));
%设置过零检测的门限。
% 寻找滤波后的过零点： +- 和-+表示水平方向从左到右和从右到左过零。
%[+-]' 和[-+]'表示垂直方向从上到下和从下到上过零。
%这里我们选择边缘点为值为负的点。
  [rx,cx]=find(b(rr,cc)<0&b(rr,cc+1)>0...
    &abs(b(rr,cc)-b(rr,cc+1))>thresh);   %[- +]的情况
  e((rx+1)+cx*m)=1;
  [rx,cx]=find(b(rr,cc-1)>0&b(rr,cc)<0...
    &abs(b(rr,cc-1)-b(rr,cc))>thresh);   %[+ -]的情况
  e((rx+1)+cx*m)=1;
  [rx,cx]=find(b(rr,cc)<0&b(rr+1,cc)>0...
    &abs(b(rr,cc)-b(rr+1,cc))>thresh);   %[- +]'的情况
  e((rx+1)+cx*m)=1;
  [rx,cx]=find(b(rr-1,cc)>0&b(rr,cc)<0 ...
    &abs(b(rr-1,cc)-b(rr,cc))>thresh);   %[+ -]'的情况
  e((rx+1)+cx*m)=1;
%某些情况下 LoG 滤波结果可能正好为 0，下面考虑这种情况：
  [rz,cz]=find(b(rr,cc)==0);
  if~isempty(rz)
    %寻找滤波后的过零。
%+0-和-0+表示水平方向从左到右和从右到左过零。
%[+0-]'和[-0+]'表示垂直方向从上到下和从下到上过零。
    % 边缘正好位于滤波值为零点上。
    zero=(rz+1)+cz*m;   % 零点的线性坐标。
    zz=find(b(zero-1)<0&b(zero+1)>0...
      &abs(b(zero-1)-b(zero+1))>2*thresh);%[-0+]'情况
    e(zero(zz))=1;
    zz=find(b(zero-1)>0&b(zero+1)<0...
      &abs(b(zero-1)-b(zero+1))>2*thresh);%[+0-]'情况
    e(zero(zz))=1;
    zz=find(b(zero-m)<0&b(zero+m)> 0 ...
      &abs(b(zero-m)-b(zero+m))>2*thresh);%[-0+]情况
    e(zero(zz))=1;
    zz=find(b(zero-m)>0&b(zero+m)<0 ...
      &abs(b(zero-m)-b(zero+m))>2*thresh);%[+0-]情况
    e(zero(zz))=1;
  end
```

MATLAB 图像处理工具箱中已经用 edge 函数提供了该算法，对 rice.tif 图像采用拉普拉斯高斯算子进行边缘提取，结果如图 8.5 所示。

从图 8.5 可以看出，边缘提取的结果要优于 Roberts 算子和 Sobel 算子，特别是边缘比较完整，位置比较准确。

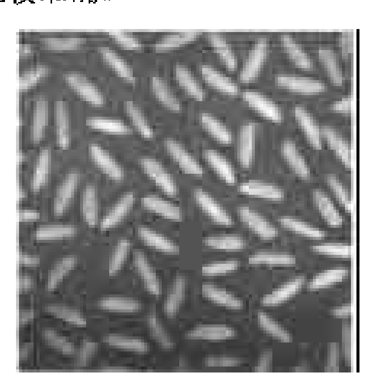



图 8.5 原始图像及LoG算子进行边缘提取的结果

8.1.3 canny 法

canny 边缘检测是一种比较新的边缘检测算子，具有很好的边缘检测性能，得到了越来越广泛的应用。canny 边缘检测法利用高斯函数的一阶微分，它能在噪声抑制和边缘检测之间取得较好的平衡。具体步骤如下：

(1) 用高斯滤波器来对图像滤波，可以去除图像中的噪声。

(2) 用高斯算子的一阶微分对图像进行滤波，得到每个像素梯度的大小 $|G|$ 和方向 θ。

$$|G| = \left[\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2\right]^{\frac{1}{2}}$$

$$\alpha = \tan^{-1}\left[\frac{\partial f}{\partial y}\bigg/\frac{\partial f}{\partial x}\right]$$

f 为滤波后的图像。

(3) 对梯度进行“非极大抑制”。

梯度的方向可以被定义为属于 4 个区之一，各个区用不同的邻近像素用来进行比较，以决定局部极大值。这 4 个区及其相应的比较方向如表 8.1 所示。

表 8.1 4个区入其相应的比较方向

4	3	2
1	x	1
2	3	4

例如，如果中心像素 x 的梯度方向属于第 4 区，则把 x 的梯度值与它的左上和右下相邻像素的梯度值比较，看 x 的梯度值是否是局部极大值。如果不是，就把像素 x 的灰度设为 0，这个过程称为“非极大抑制”。

(4) 对梯度取两次阈值得到两个阈值 T1 和 T2，T1=0.4*T2。我们把梯度值小于 T1 的像素的灰度设为 0，得到图像 1。然后把梯度值小于 T2 的像素的灰度设为 0，得到图像 2。由于图像 2 的阈值较高，去除了大部分噪声，但同时也损失了有用的边缘信息。而图像 1 的阈值较低，保留了较多的信息。我们可以以图像 2 为基础，以图像 1 为补充来连接图像的边缘。

(5) 连接边缘的具体步骤如下：

① 对图像 2 进行扫描，当遇到一个非零灰度的像素 P 时，跟踪以 P 为开始点的轮廓线，直到该轮廓线的终点 Q。

② 考察图像 1 中与图像 2 中 Q 点位置对应的点 Q'的 8-邻近区域。如果在 Q'点的 8-邻近区域中有非零像素 R'存在，则将其包括到图像 2 中，作为点 R。从 R 开始，重复第①步，直到我们在图像 1 和图像 2 中都无法继续为止。

③ 当完成对包含 P 的轮廓线的连接之后，将这条轮廓线标记为已访问。回到第①步，寻找下一条轮廓线。重复步骤①、②、③，直到图像 2 中找不到新轮廓线为止。

下面的代码可以实现 Canny 算子边缘检测功能：

```
function e=canny_edge(a,sigma)
%该函数实现 Canny 算子边缘检测。
%输入图像 a，标准差 sigma，输出边缘 e。

[m,n]=size(a);
rr=2:m-1;cc=2:n-1;
e=repmat(logical(uint8(0)),m,n);
%初始化边缘矩阵。

GaussianDieOff=.0001;        %设定高斯函数消失门限。
PercentOfPixelsNotEdges=.7;  % 用于计算边缘门限。
ThresholdRatio=.4;           %设置两个门限的比例。
%首先设计高斯滤波器和它的微分。
pw=1:30;
%设定滤波器宽度。
ssq=sigma*sigma;
%计算方差
width=max(find(exp(-(pw.*pw)/(2*sigma*sigma))>GaussianDieOff));
%计算滤波算子宽度。
t=(-width:width);
len=2*width+1;
t3=[t-.5;t;t+.5];
%对每个像素左右各半个像素位置的值进行平均。
gau=sum(exp(-(t3.*t3)/(2*ssq))).'/(6*pi*ssq);
%一维高斯滤波器
dgau=(-t.* exp(-(t.*t)/(2*ssq))/ssq).';
% 高斯滤波器的微分
ra=size(a,1);
ca=size(a,2);
ay=255*a;ax=255*a';
h=conv(gau,dgau);
%利用高斯函数滤除噪声和用高斯算子的一阶微分对图像滤波合并为一个算子。
```

```
ax=conv2(ax,h,'same').';
%产生 x 方向滤波。
ay=conv2(ay,h,'same');
%产生 y 方向滤波。
mag=sqrt((ax.*ax)+(ay.*ay));
%计算滤波结果的幅度。
magmax=max(mag(:));
if magmax>0
   mag=mag/magmax;
% 对滤波幅度进行归一化。
end
%下面根据滤波幅度的概率密度计算滤波门限。
[counts,x]=imhist(mag,64);
%计算滤波结果的幅度的直方图。
highThresh=min(find(cumsum(counts)>PercentOfPixelsNotEdges*m*n))/64;
%通过设定非边缘点的比例来确定高门限。
lowThresh=ThresholdRatio*highThresh;
%设置低门限为高门限乘以比例因子。
thresh=[lowThresh highThresh];
%下面进行非极大抑制。
%大于高门限的点归于强边缘图像。
%小于低门限的点归于弱边缘图像。
idxStrong=[];
for dir=1:4
   idxLocalMax=cannyFindLocalMaxima(dir,ax,ay,mag);
   idxWeak=idxLocalMax(mag(idxLocalMax)>lowThresh);
   e(idxWeak)=1;
   idxStrong=[idxStrong;idxWeak(mag(idxWeak)>highThresh)];
end
cstrong=rem(idxStrong-1,m)+1;
cstrong=floor((idxStrong-1)/m)+1;
e=bwselect(e,cstrong,rstrong,8);
%通过形态学算子将两幅图像的边缘进行连接。
e=bwmorph(e,'thin',1);
%对提取的边缘利用形态学算子细化。
```

上面代码中的 cannyFindLocalMaxima 实现“非极大抑制”功能，其代码如下：

```
function idxLocalMax = cannyFindLocalMaxima(direction,ix,iy,mag);
%输入 direction 为 4 个方向，ix 为图像在 x 方向滤波结果。
%iy 为图像在 y 方向滤波结果，mag 为滤波幅度。
[m,n,o] = size(mag);
%根据梯度幅度确定各点梯度的方向，并找出四个方向可能存在边缘点的坐标。
switch direction
case 1
   idx=find((iy<=0&ix>-iy)|(iy>=0&ix<-iy));
case 2
   idx=find((ix>0&-iy>=ix)|(ix<0&-iy<=ix));
case 3
   idx=find((ix<=0&ix>iy)|(ix>=0&ix<iy));
case 4
```

```
      idx=find((iy<0&ix<=iy)|(iy>0&ix>=iy));
   end
   %去除图像边界以外点。
   if~isempty(idx)
      v=mod(idx,m);
      extIdx=find(v==1|v==0|idx<=m|(idx>(n-1)*m));
      idx(extIdx)=[];
   end
   %求出可能的边缘点的滤波值。
   ixv=ix(idx);
   iyv=iy(idx);
   gradmag=mag(idx);
   %计算4个方向的梯度幅度。
   switch direction
   case 1
      d=abs(iyv./ixv);
      gradmag1=mag(idx+m).*(1-d)+mag(idx+m-1).*d;
      gradmag2=mag(idx-m).*(1-d)+mag(idx-m+1).*d;
   case 2
      d=abs(ixv./iyv);
      gradmag1=mag(idx-1).*(1-d)+mag(idx+m-1).*d;
      gradmag2=mag(idx+1).*(1-d)+mag(idx-m+1).*d;
   case 3
      d=abs(ixv./iyv);
      gradmag1=mag(idx-1).*(1-d)+mag(idx-m-1).*d;
      gradmag2=mag(idx+1).*(1-d)+mag(idx+m+1).*d;
   case 4
      d=abs(iyv./ixv);
      gradmag1=mag(idx-m).*(1-d)+mag(idx-m-1).*d;
      gradmag2=mag(idx+m).*(1-d)+mag(idx+m+1).*d;
   end
   idxLocalMax=idx(gradmag>=gradmag1&gradmag>=gradmag2);
   %进行"非极大抑制"。
```

MATLAB 图像处理工具箱中的 edge 函数也提供了 Canny 算法的功能，其对 rice.tif 图像的边缘提取结果，如图 8.6 所示。

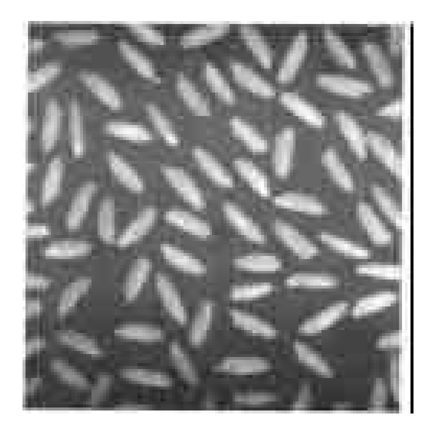



图 8.6 Canny算子提取边缘结果

从图上可以看出，Canny 算子提取的边缘十分完整，而且边缘的连续性很好，效果优

于其他算子。这是因为它进行了“非极大值抑制”和形态学连接操作的结果。

8.2 直线提取

因为直线通常对应重要的边缘信息，直线提取是计算机视觉中一项非常重要的技术。例如车辆自动驾驶技术中道路的提取需要有效地提取直的道路边缘，也就是提取获取的图像中的直线；而航空照片分析中，直线更是对应于重要的人造目标的边缘。因此把直线单独提取抽出来进行研究很有意义。而且，由于直线具有不同于一般曲线的特征，因此它的提取方法也与一般的边缘检测方法不同。

8.2.1 Hough 变换法

利用 Hough 变换法提取直线是一种变换域提取直线的方法，它把直线上点的坐标变换到过点的直线的系数域，巧妙地利用了共线和直线相交的关系，使直线的提取问题转化为计数问题。Hough 变换提取直线的主要优点是受直线中的间隙和噪声影响较小。

平面 O-xy 上的直线方程为：

$$y=ux+v$$

其中 u 和 v 分别为直线的斜率和截距。对于给定的一条直线，对应一个数对(u,v)，反之，如果给定一个数对(u,v)，则对应一条直线 y=ux+v。即平面 O-xy 上的直线 y=ux+v 和 O-uv 与平面的一个数对(u,v)构成一一对应。这个关系称为 Hough 变换。同理，O-uv 平面上的一条直线 v=-xu+y 与 O-xy 上的点(x,y)也是一一对应的，如图 8.7 所示。

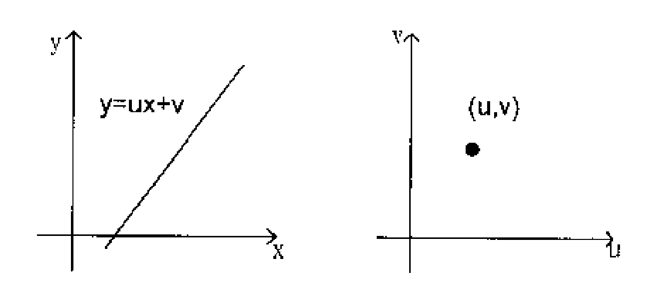


图 8.7　O-xy平面上直线与O-uv平面上点的对应关系

因此，如果 O-xy 平面上有一条直线 y=ux+v，那么它上面的每一个点都对应于 O-uv 平面上的一条直线，这些直线相交于一点(u,v)。利用这个重要性质可以检测共线点。

注意到直线的斜率可能会接近无穷大，为了使变换域有意义，需要采用直线方程的法线式表示：

$$x\cos\theta+y\sin\theta=\rho$$

式中 ρ 是直线到坐标系原点的距离，θ 是直线法线与 X 轴的夹角。于是，坐标平面 O–xy 中的一条直线和坐标平面 O–$\rho\theta$ 中的一点一一对应，O–xy 坐标平面中的一点和 O–$\rho\theta$ 中的一条曲线一一对应，而且容易知道 O–xy 中的共线点所对应的 O–$\rho\theta$ 中的曲线交于一点，

如图 8.8 所示。

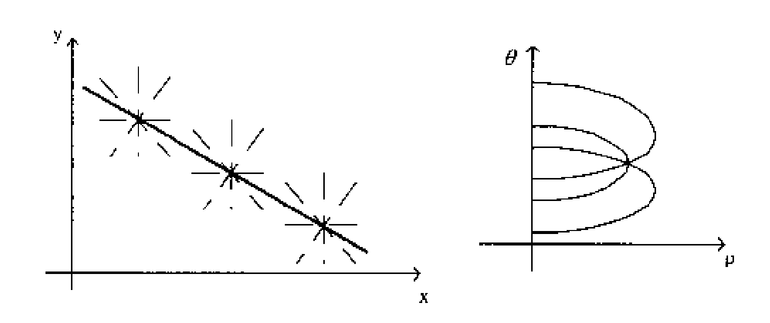


图 8.8 共线点对应的曲线相交于一点

可以知道 O-xy 中直线上的各点对应着 O-ρ″的一个点。如果对过这一点的曲线进行计数，结果会是比较大的数值。因此我们可以根据精度 O-ρ″平面划分为等间隔的小直网格，这个直网格对应一个记数阵列。对于 O-xy 平面中的每一点，按上面介绍的原理在 O-ρ″平面中画出它对应的曲线，凡是这条曲线所经过的小格，对应的记数阵列元素加 1。于是记数阵元的数值等于共线的点数。当我们检测直线时，对应于大记数的小格，通过它的那些曲线所对应的 O-xy 平面的各点接近于共线，而通过小记数的小格的曲线的对应点则认为是孤立点，不构成直线，应该去除。利用这个方法检测直线称为 Hough 变换直线检测方法。

利用 Hough 变换检测直线的原理和方法也可以检测其他的能用解析式表示的参数较少的曲线，如圆、椭圆及抛物线。比如可以利用 Hough 变换检测圆。为了减少变换空间维数而又保持一般性，这里设圆的半径 r 是已知常数。O-xy 平面中一个半径为 r 的圆的表达式如下：

$$(x-a)^2+(y-b)^2=r^2$$

在变换空间 O-ab 平面中有一点(a,b)与之一一对应。对于 O-ab 平面中的一个圆 $(x-a)^2+(y-b)^2=r^2$，在 O-xy 中有一点(x,y)与之一一对应。在 O-xy 中所有共圆的点，它们所对应的 O-ρ″中的圆交于一点。运用这个性质采用上述的方法即可检测 O-xy 中的共圆点(圆半径为 r)。事实上，还可以用上述原理检测形状复杂而不能用解析式表示的目标。这种方法称为广义 Hough 变换方法。

下面给出 hough 变换的代码。

```
function hough(x)
%该函数实现 hough 变换提取直线的功能。
%输入图像 x，运行之后直接画出直线。
[m,n]=size(x);
%求出图像大小。
md=round(sqrt(m^2+n^2));
%确定网格的最大区域。
ma=180;
ruthta=zeros(md,ma);
%产生计数矩阵。
ruthx=cell(1,1);
%cell 数组相当于 c 语言中的指针，可动态改变大小。
for i=1:md
```

```
    for j=1:ma
      ruthx{i,j}=[];
    end
end
%产生空网格。
for i=5:m-4
    for j=5:n-4
      if bw(i,j)==1
        for k=1:ma
          ru=round(abs(j*cos(I(k))+I*sin(I(k))));
          %根据直线的法线式表示，计算出 %平面上不同点的 hough 变换值。
          ruthta(ru+1,k)=ruthta(ru+1,k)+1;
          %将 hough 变换值相应位置的计数值加 1。
          ruthx{ru+1,k}=[ruthx{ru+1,k},[i,j]'];
         %记录 hough 变换值相应位置对应的点的坐标。
        end
      end
    end
end
figure(1)
bw=ones(size(bw));
imshow(bw);
N=1; %这里假设图像上有一条主要直线。
for i=1:N
%这里假定图像上有 N 条主要直线。
   [y1,col1]=max(ruthta);
   [y2,row]=max(y1);
     %求出 hough 变换值最大值的坐标。
col=col1(row);
   ruthta(col,row)=0;
   %为避免重复计算，将计算过的点置为 0。
   N=3;
   if col-N>0&col+N<md&row-N>0&row+N<ma
      ruthta(col-N:col+N,row-N:row+N)=0;
   end
   nds=ruthx{col,row};
      pline=draw_l(nds);
end
function pline=draw_l(im)
%此函数实现根据在一条直线上点的坐标,计算出直线参数，并画出直线的功能。
y=im(1,:);
x=im(2,:);
mx=max(x); nx=min(x);
%确定直线在 x 方向的范围。
my=max(y); ny=min(y);
%确定直线在 y 方向的范围。
cx=mean(x); cy=mean(y);
%计算出直线的中心点坐标。
xx=x-cx; yy=y-cy;
a=sum(xx.^2)-sum(xx)^2;
b=sum(xx.*yy)-sum(xx)*sum(yy);
```

```
c=sum(yy.^2)-sum(yy)^2;
Vs=(a+c)/2+sqrt((a-c)^2/4+b^2);
%利用最小二乘拟合直线，求出与直线斜率有关的 Vs。
if abs((Vs-a)/(b+eps))<=1
   my=floor(cy+(Vs-a)/(b+eps)*(mx-cx));
   ny=floor(cy+(Vs-a)/(b+eps)*(nx-cx));
else
   mx=floor(cx+b/(Vs-a+eps)*(my-cy));
   nx=floor(cx+b/(Vs-a+eps)*(ny-cy));
end
%为避免斜率大于 1 时计算端点坐标误差过大，根据斜率值分别确定最大的 x 坐标或者 y 坐标。
line([nx,mx],[ny,my]);
%画出直线。
pline=[[nx,ny]',[mx,my]'];
```

Hough 变换的缺点是运算量太大，而且由于不考虑各点之间的距离信息，因此容易将不属于直线的点也连接到直线上，即产生所谓的过连接现象。实际中，为了克服运算量大的缺点，通常可以把图像划分为小块，对各块图像利用 Hough 变换提取直线，然后再将各直线连接起来。

下面是利用 Hough 变换的方法提取直线的例子。为了显示清楚，我们只提取了一条主要直线，用红线表示。可以看出红线所在的直线上像素最多。注意到红线的端点超出了实际的位置，这就是所谓的过连接现象，如图 8.9 所示。

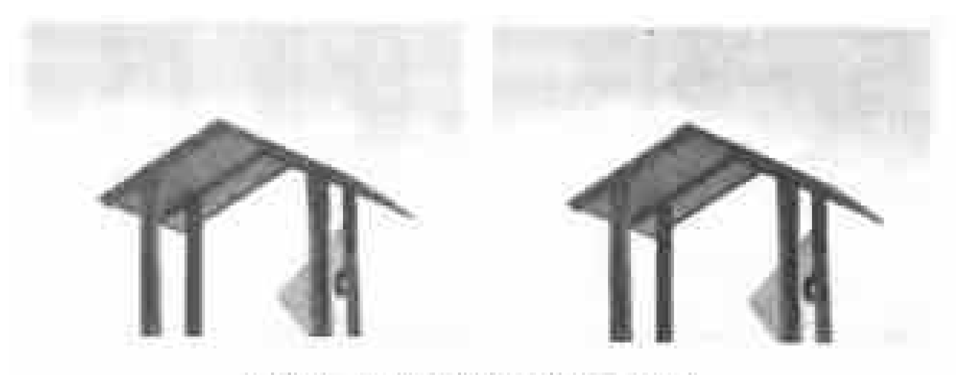


图 8.9 Hough变换法提取的直线结果

8.2.2 相位编组法

前面提到的边缘提取算法，比如梯度模算子法，拉普拉斯高斯算子法等都有一个共同的特点，即利用的是梯度的幅度信息。如果某一点的梯度模超过了预先设定的门限，则认为存在边缘。它的基本原理是，认为边缘是灰度发生突变的地方。在图像的灰度缓慢变化的情况下，利用这些方法效果不是很好。

梯度相位法采用的是另一种思路，它认为边缘不只是在灰度发生突变的地方存在，如果某个区域，灰度沿某个方向缓慢变化，这个区域也存在边缘，只不过可以认为边缘比较粗。

因此，梯度的方向也是非常重要的信息，如果在某个区域，各点的梯度的方向相同或者相近，则这个区域很可能存在边缘。相位编组法的思想是求出图像中各个像素的梯度相

位，将相邻的方向相同的点编为一个直线支撑区，然后根据支撑区内像素的坐标求出直线的方程，即可将直线提取出来。图 8.10 中箭头代表梯度方向，相同方向的箭头被编为一组，构成直线支撑区。

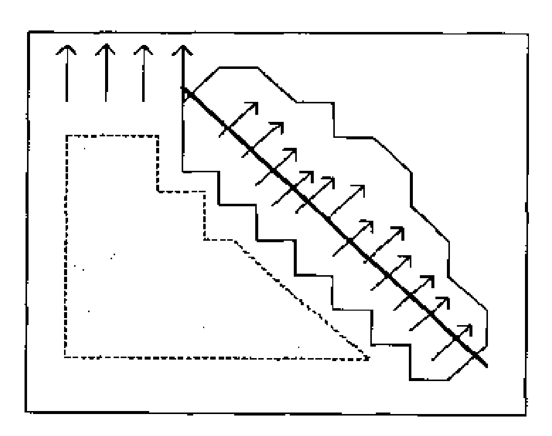


图 8.10　直线支撑区示意图

利用相位编组法提取直线可以按照下面的步骤进行：

(1) 求出图像上各像素的梯度的方向，计算方法为：

$$\alpha = \tan^{-1}\left[\frac{\partial f}{\partial y}\bigg/\frac{\partial f}{\partial x}\right]$$

(2) 根据各像素梯度的方向将它们划分为 8 个区域，如图 8.11 所示。

(3) 提取出图像上的边缘点，这可以用梯度算子法或者 LoG 算子和 Canny 算子求出。

(4) 从这些边缘点出发，将梯度方向在一个区域而且相互之间临近的点划分为一个编组区。这可以通过从边缘点出发在其 8 个邻域内搜索方向一致的点来实现。

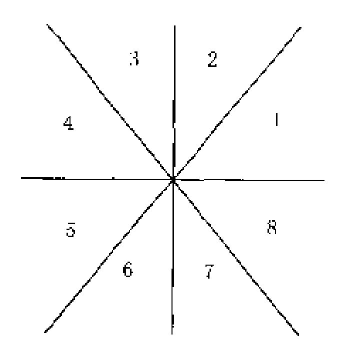


图 8.11　相位编组 8 个方向的划分方法

(5) 用各相位编组区的点的灰度拟合灰度表面。

(6) 利用直线编组区的点的坐标和灰度表面求出直线的特性，如端点坐标、长度、宽度以及直线度等。

由于受噪声影响，像素的相位容易发生突变，使得使用相位编组的方法时，各直线的编组区容易发生断裂。在实际应用中，还要进行直线连接操作，即将在同一直线上但是不连续的线段合并成一条直线。

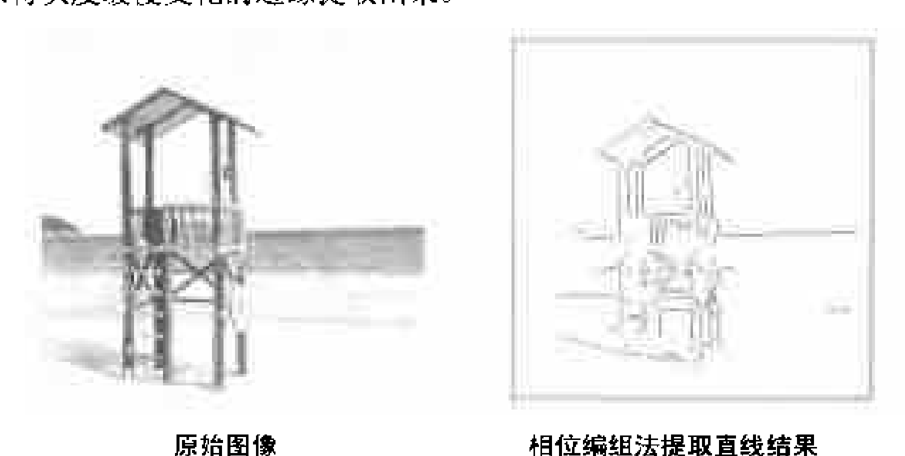
下面给出相位编组法的 MATLAB 代码：

```
function gradephase(x)
%该函数实现相位编组法提取直线功能，输入图像 x，输出直接画出直线。
[m,n]=size(x);
bw=edge(x,'sobel');
%首先提取边缘点。
gy=x(1:m-1,1:n-1)-x(2:m,1:n-1);
```

```
%矩阵在 y 方向的差分代替微分。
gx=x(1:m-1,1:n-1)-x(1:m-1,2:n);
%矩阵在 x 方向的差分代替微分。
g=gy./(gx+eps);
%求出各像素梯度的正切值。
ph=atan(g)+(sign(gx)<0 & sign(gy)>0)*pi+...
   (sign(gx)<0 & sign(gy)<0)*pi+(sign(gx)>0 & sign(gy)<0)*2*pi+...
   (sign(gx)==0)*pi;
%利用矩阵运算提高相位计算效率。
grdgp=floor(ph/pi*4)+1;
%对各像素的相位进行分组。
cn=0;   s=[];
ln_spt=cell(1,1);
%cell 数组类似 c 语言的指针，可以动态改变大小。
%用来存放直线编组区内的像素坐标。
pline=[];
for i=2:m-1
   for j=2:n-1
      if bw(I,j)~=0&grdgp(i,j)~=0    %搜索边缘点。
         ph_cp=grdgp(i,j);   grdgp(I,j)=0;  bw(I,j)=0;
         cn=cn+1;  p=[I,j]';
         s=[s,p];  ln_spt{cn}=[p];
%为了在 8 个邻域进行搜索，采用入栈和出栈操作。
         while ~isempty(s)
            [cs,rs]=size(s);
            ps=s(:,rs);  s=s(:,1:rs-1);
            col=ps(1,1); row=ps(2,1);
            if  legal(col+1,row,m,n)&grdgp(col+1,row)==ph_cp
s=[s,[col+1,row]'];  ln_spt{cn}=[ln_spt{cn},[col+1,row]'];
               bw(col+1,row)=0;    grdgp(col+1,row)=0;
            end
            if legal(col+1,row+1,m,n)&grdgp(col+1,row+1)==ph_cp
s=[s,[col+1,row+1]'];  ln_spt{cn}=[ln_spt{cn},[col+1,row+1]'];
               bw(col+1,row+1)=0;  grdgp(col+1,row+1)=0;
            end
            if legal(col,row+1,m,n)&grdgp(col,row+1)==ph_cp
               s=[s,[col,row+1]'];  ln_spt{cn}=[ln_spt{cn},[col,row+1]'];
               bw(col,row+1)=0;    grdgp(col,row+1)=0;
            end
            if legal(col-1,row+1,m,n)&grdgp(col-1,row+1)==ph_cp
              s=[s,[col-1,row+1]']; ln_spt{cn}=[ln_spt{cn},[col-1,row+1]'];
               bw(col-1,row+1)=0;  grdgp(col-1,row+1)=0;
            end
            if  legal(col-1,row,m,n)&grdgp(col-1,row)==ph_cp
               s=[s,[col-1,row]'];  ln_spt{cn}=[ln_spt{cn},[col-1,row]'];
               bw(col-1,row)=0;     grdgp(col-1,row)=0;
            end
            if  legal(col-1,row-1,m,n)&grdgp(col-1,row-1)==ph_cp
              s=[s,[col-1,row-1]'];
ln_spt{cn}=[ln_spt{cn},[col-1,row-1]'];
               bw(col-1,row-1)=0;  grdgp(col-1,row-1)=0;
```

```
            end
            if  legal(col,row-1,m,n)&grdgp(col,row-1)==ph_cp
             s=[s,[col,row-1]'];  ln_spt{cn}=[ln_spt{cn},[col,row-1]'];
               bw(col,row-1)=0;   grdgp(col,row-1)=0;
            end
            if  legal(col+1,row-1,m,n)&grdgp(col+1,row-1)==ph_cp
              s=[s,[col+1,row-1]']; ln_spt{cn}=[ln_spt{cn},[col+1,row-1]'];
               bw(col+1,row-1)=0;  grdgp(col+1,row-1)=0;
            end
         end
%以上判断是在 8 个方向搜索相同方向的像素。
         if length(ln_spt{cn})<=10
            ln_spt{cn}=[];
            cn=cn-1;
%滤除短线段。
         else
            plne=draw_l(ln_spt{cn});
%实现画线功能，代码与 Hough 变换中的代码相同。
            pline=[pline,plne];
            c(i)=length(ln_spt{cn});
         end
      end
   end
end
```

图 8.12 是一幅利用相位编组法提取直线得到的结果，可以看出，利用相位编组法提取直线，可以将灰度缓慢变化的边缘提取出来。



原始图像　　　　相位编组法提取直线结果

图 8.12　原始图像及提取边缘后的结果

8.3　基于灰度分割

图像分割是图像处理与机器视觉的基本问题之一，其任务是把图像划分成互不交迭区域的集合。这些区域的划分是有实际意义的，它们或者代表不同的物体，或者代表物体的不同部分。图像分割的一个难点在于，在划分之前，不一定能够确定图像区域的数目。

图像分割的用途非常广泛，几乎涉及有关图像处理的所有领域，应用于各种类型的图像。例如，在医学应用中，脑部核磁共振(MR)图像分割成灰质、白质、脑脊髓等脑组织和其他非脑组织区域等；在交通图像分析中，把车辆目标从背景中分割出来；在面向对象的图像压缩和基于内容的图像数据库查询中，将图像分割成不同的对象区域作为基元进行处理；在遥感云图中利用纹理的差别分割不同背景分布的云系等。在这些应用中，分割通常用来对图像进行进一步的分析、识别及压缩编码等，分割的准确性直接影响后续任务的有效性，因此具有十分重要的意义。

按照通用的分割定义，分割出的区域需同时满足均匀性和连通性的条件。其中均匀性是指该区域中的所有像素点都满足基于灰度、纹理、颜色等特征的某种相似性准则，而连通性是指在该区域内任意两点存在相互连通的路径。

设 F 表示一幅图像中所有像素的集合，P(•)是某个均匀性的假设，分割就是把 F 划分成若干子集(S1,S2,…,Sn)，其中各子集构成一个空间连通区域，且满足以下条件：

$$\bigcup_{i=1}^{n} S_i = F \quad 且\ S_i \bigcap S_j = \Phi, i \neq j$$

$$P(\bullet)满足\ P(S_i) = true, \forall i$$

$$P(S_i \bigcap S_j) = false，若\ S_i\ 与\ S_j\ 在空间相邻$$

完全符合上述定义的分割计算十分复杂，目前大部分研究都是针对某一类型图像或者某一具体应用的分割，人们仍在对图像分割的通用方法和策略进行研究。

通常情况下，利用目标区域和背景区域在灰度方面的差异，可以实现对图像的分割，即基于灰度的图像分割。

8.3.1　灰度门限法

设图像 f(x,y)的灰度范围属于[z1,z2]，根据一定的经验及知识确定一个灰度的门限，或者根据一定准则确定[z1,z2]的一个划分 Z1，Z2，其中 Z1 代表目标，Z2 代表背景。根据像素的灰度属于这个划分的哪个部分来将其分类，称为灰度门限法，即：

如果 f(x,y)属于 Z1 ，判断(x,y)像素属于目标。

如果 f(x,y)属于 Z2，则判断 (x,y)像素属于背景。

根据划分方法的不同，可以将灰度门限法分为：

1.　直接门限法

如果在目标区域和背景区域的内部，像素间的灰度都基本一致，而目标和背景区域的像素灰度有一定差异，可以根据灰度不同直接设定灰度门限进行分割。例如对于含有细胞的医学图像，细胞的灰度通常比背景低得多，这时可以根据某种准则求出一个门限，当像素的灰度值低于门限时，判断像素属于目标，即可将细胞提取出来。

2. 间接门限法

在有些情况下，如果对图像作一些必要的预处理然后再运用门限法，可以有效地实现图像分割。例如，图像只有黑色和白色两个灰度，但白色象素在目标区域中出现的概率比在背景区域中出现的概率大，即目标区域的平均灰度高于背景区域，可以先对图像进行邻域平均运算，再对新图运用门限法进行分割。

又如，图像中目标区域灰度变化剧烈，而背景区域变化平缓，可以先对原图像进行拉氏运算，突出目标区域的特征，然后对新图使用邻域平均技术，最后再用门限法实行有效地分割。

3. 多门限法

有时候一幅图像含有两个以上不同类型的区域，用直接或者间接单门限的方法无法将两个以上的目标区域提取出来，这时可以使用多个门限将这些区域划分开。对于只有两个类型区域的图像有时也要使用多门限的方法，例如图像是在照度不均条件下摄取的，若使用单一门限对整幅图像进行分割，可能发生在图像的一边能精确地把目标和背景分开，而在另一边可能把太多的背景点当作目标点保留下来的情况，或者正好相反，得不到好的分割结果。在这种情况下，可以运用同态滤波技术校正灰度，然后再用单一门限进行分割，这相当于对图像进行预处理的间接门限法。同时还可以把图像分成若干个子图，各子图像中目标和背景相对差别比较分明，可以分别对每个子图进行单门限分割，再将分割结果综合起来。

使用双门限法也可以提高对两类区域的图像的分割精度。方法是设置一高一低两个门限 t1、t2，不妨设 t1<t2。选择 t2 使有些目标点的灰度大于 t2，选择 t1 应使每个目标点的灰度均高于 t1。在进行分割时，把灰度大于 t2 的像点作为“核心”目标点，对于灰度超过 t1 的象素再根据它和核心目标点的距离判断，如果相邻则将其当作目标点。这种分割方式除了利用灰度信息还使用了空间距离信息，因此分割效果较好。

8.3.2 灰度门限的确定

分割门限选择的准确性直接影响分割的精度及图像描述分析的正确性。门限选得太高，容易把大量的目标判为背景，定的太低又会把大量的背景判为目标。因此正确分割门限是很重要的。通常采用根据先验知识确定门限，或者利用灰度直方图特征和统计判决方法确定灰度分割门限。

如果已知被处理图像的一些先验信息，可以利用试探的方法确定门限。例如已知一幅文字图像中文字占有的比例为 x，可以选择这样一个门限，使得对图像进行门限化处理后，灰度小于门限的象素的数目约占总象素的百分之 x。

很多时候，我们并没有关于图像充分的先验知识，因此只能从图像本身的特征出发来确定门限。图像的统计特性可以提供分割的依据。利用灰度直方图特征确定灰度分割门限的原理是：如果图像所含的目标区域和背景区域大小可比，而且目标区域和背景区域在灰度上有一定的差别，那么该图像的灰度直方图会呈现双峰一谷状：其中一个峰值对应于目

标的中心灰度，另一个峰值对应于背景的中心灰度。由于目标边界点较少且其灰度介于它们之间，所以双峰之间的谷点对应着边界的灰度，可以将谷点的灰度作为分割门限。

由于直方图的参差性，实际上寻找谷值并不是一个简单的过程，需要设计一定的准则进行搜索。假设图像的直方图为 h，确定直方图谷点的位置方法之一是通过搜索找出直方图的两个最大的局部最大值，设它们的位置是 Z1 和 Z2，并且要求这两点距离大于某个设定的距离，然后求 Z1 和 Z2 之间直方图最低点 Zm，用 h(Zm)/min (h(Z1),h(Z2))测度直方图的平坦性，若这个值很小，则表示直方图是双峰一谷状，可将 Zm 作为分割门限。

寻找灰度门限也可以用一个解析函数来拟合直方图双峰间的部分，然后再用函数求极值的方法找出这个解析函数最小值的位置作为谷点。例如可用二次曲线：

$$y = ax^2 + bx + c$$

来拟合双峰之间的直方图，求得拟合系数，则 $x = -\frac{b}{2a}$ 可作为分割门限。

类似的也可以使用两个高斯函数拟合直方图的两个峰，然后求出两个高斯函数的交点来确定门限。

需要指出的是，由于直方图是各灰度的像素统计，未考虑图像其他方面的知识，只靠直方图分割的结果有可能是错误的。即使直方图是典型的双峰一谷特性，这个图像也未必含有和背景有反差的目标。例如一幅左边是黑色而右边是白色的图像和一幅黑白像点随机分布的图像具有相同的直方图，但是后者就不包含有意义的目标。

另外还可以利用统计判决确定门限，比如利用最小误判概率准则确定分割的最佳门限。

设图像含有目标和背景，目标点出现的概率为 θ，其灰度分布密度为 p(x)，背景点的灰度分布密度为 q(x)。按照概率论理论，这幅图像的灰度分布密度函数为：

$$s(x) = \theta p(x) + (1-\theta)qx$$

假设我们根据灰度门限 t 对图像进行分割，并将灰度小于 t 的像点作为背景点，灰度大于 t 的像素作为目标点，于是将目标点误判为背景点的概率为：

$$\varepsilon_{12} = \int_{-\infty}^{t} p(x)dx$$

把背景点误判为目标点的概率为：

$$\varepsilon_{21} = \int_{t}^{+\infty} q(x)dx$$

按照总误判概率最小准则，我们选取的门限 t 应使下式最小：

$$\varepsilon = \theta\varepsilon_{12} + (1-\theta)\varepsilon_{21}$$

根据函数求极值的方法，对 t 求导并令结果为零，有：

$$\theta p(t) + (1-\theta)q(t) = 0$$

这是典型的统计判决方法。另外还可以采用最大后验概率方法、最小误判风险方法等。已知 θ、p(x)及 q(x)则可以求解出最佳门限 t。而且对于正态分布、瑞利分布、对数正态分

布，最佳门限 t 是容易求解的。

在不知道图像灰度分布的情况下，还可以采用模式识别中最大类间方差准则确定分割的最佳门限。其基本思想是对像素进行划分，通过使划分得到的各类之间的距离达到最大，来确定合适的门限。

设图像 f 中，灰度值为 I 的像素的数目是 n_i，总像素数为：

$$N = \sum_{i=1}^{L} n_i$$

各灰度出现的概率为：

$$p_i = \frac{n_i}{N}$$

设以灰度 k 为门限将图像分为两个区域，灰度为 1~k 的像素和灰度为 k+1~L 的像素分别属于区域 A 和 B，则区域 A 和 B 的概率分别为：

$$\omega_A = \sum_{i=1}^{k} p_i \text{ 和 } \omega_B = \sum_{i=k+1}^{L} p_i$$

为简便起见，定义 $\omega_A = \omega(k)$。

区域 A 和 B 的平均灰度为：

$$\mu_A = \frac{1}{\omega_A}\sum_{i=1}^{k} i * p_i \overset{\Delta}{=} \frac{\mu(k)}{\omega(k)} \quad \text{和 } \mu_B = \frac{1}{\omega_B}\sum_{i=k+1}^{L} i * p_i \overset{\Delta}{=} \frac{\mu - \mu(k)}{1-\omega(k)}$$

其中 μ 为全图的平均灰度，

$$\mu = \sum_{i=1}^{L} i * p_i = \omega_A \mu_A + \omega_B \mu_B$$

两个区域的方差为 $\sigma^2 = \omega_A(\mu_A - \mu)^2 + \omega_B(\mu_B - \mu)^2$

$$= \frac{[\mu\omega(k) - \mu(k)]^2}{\omega(k)[1-\omega(k)]}$$

按照最大类间方差的准则，从 1 至 L 改变 k，并计算类间方差，使上式最大的 k 即是区域分割的门限。

下面是最大方差法计算灰度分割门限代码：

```
function th=thresh_md(a);
%该函数实现最大方差法计算分割门限。
%输入参数为灰度图像，输出为灰度分割门限。
count=imhist(a);
%返回图像矩阵 a 各个灰度等级像素个数。
[m,n]=size(a);
N=m*n-sum(sum(find(a==0),1));
L=256;
%指定图像灰度等级为 256 级。
%d=std2((double(a)+1)/N)
%err=0.003;
count=count/N;
```

```
%计算出各灰度出现的概率。
for i=2:L
   if count(i)~=0
      st=i-1;
      break
   end
end
%找出出现概率不为 0 的最小灰度。
for i=L:-1:1
   if count(i)~=0;
      nd=i-1;
      break;
   end
end
%找出出现概率不为 0 的最大灰度。
f=count(st+1:nd+1);
p=st; q=nd-st;
%p 和 q 分别为灰度起始和结束值。
u=0;
for i=1:q;
   u=u+f(i)*(p+i-1);
   ua(i)=u;
end;
%计算图像的平均灰度。
for i=1:q;
   w(i)=sum(f(1:i));
end;
%计算出选择不同 k 值时，A 区域的概率。
d=(u*w-ua).^2./(w.*(1-w));
%求出不同 k 值时类间的方差。
[y,tp]=max(d);
%求出最大方差对应的灰度级。
th=tp+p;
```

图 8.13 是利用最大方差法求出灰度门限对一幅图像进行分割的结果，可以看出分割效果令人满意，细胞灰度较低，因此可以和背景区分开来。但是由于细胞中心灰度较亮，因此分割时被归入背景。这不是由于门限选择不正确，而是由于同一物体有不同的灰度造成的。这可以通过形态学滤波的方法来修正，形态学滤波的方法将在第 8 章进行介绍。

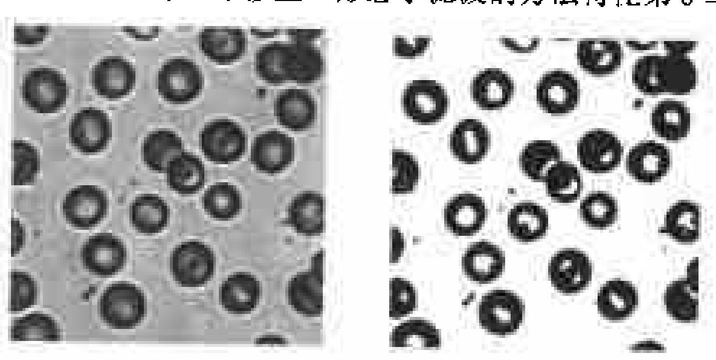


图 8.13 最大方差灰度门限法对图像进行分割的结果

8.4　分开合并算法

图像分割的方法很多，除了上面提到的灰度门限分割方法以外，还可以采用区域生长的方法，即从图像中的一些种子点开始，按照一定的相似性准则不断把邻近的象素包含进产生的区域。灰度门限法可以认为是从上到下对图像进行分开，区域生长的方法相当于从下到上对象素进行合并。如果将这两种方法结合起来对图像进行划分，就是分开合并算法。分开合并算法需要采用图像的四分树结构作为其基本数据结构，下面先对图像的四分树表示进行介绍。

8.4.1　四分树

图像除了用各个像素表示之外，还可以根据应用目的的不同，以其他方式表示，比如用四分树表示。图像的四分树可以用于图像的分割，也可以用于图像的压缩等。

首先介绍一下图像的四分树结构。四分树要求图像的大小为 2 的整数次幂。设 $N=2^n$，对于 N×N 大小的图像 f，它的金字塔数据结构是一个从 1×1 到 N×N 逐次增加的 n+1 个图像构成的序列。序列中 1×1 图像是 f 所有像素灰度的平均值构成的图像，序列中 2×2 图像是将 f 划分为 4 个大小相同且互不重叠的正方形区域，各区域的像素灰度平均值分别作为 2×2 图像相应位置上的四个像素的灰度。同样的，对已经划分的 4 个区域分别再一分为四，然后求各区域的灰度平均值将其作为 4×4 图像的像素灰度。重复这个过程，直到图像尺寸变为 N×N 为止。

也可以从相反的方向构造这个数据结构。序列中的 N×N 图像就是原始图像 f。将 f 划分为 $\frac{N}{2}\times\frac{N}{2}$ 大小相同互不重叠的正方区域，每个区域都含有 4 个像素，各区域中 4 个像素灰度平均值分别作为相应位置上 $\frac{N}{2}\times\frac{N}{2}$ 图像像素的灰度，然后再将 $\frac{N}{2}\times\frac{N}{2}$ 图像划分成 $\frac{N}{2}\times\frac{N}{2}$ 个大小相同互不重叠的正方区域，各区域中 4 个像素灰度平均值分别为相应位置上的 $\frac{N}{4}\times\frac{N}{4}$ 图像像素的灰度，依次类推。容易求出，存储这样一个图像序列所需的空间为 $\frac{4}{3}N^2\times g$，其中 g 是一个像素值的位数，即只比原图像 f 存储空间约多三分之一。采用四分树数据结构的主要优点是可以首先在较低分辨率的图像上进行需要的操作，然后根据操作结果决定是否在高分辨率图像上进一步处理、如何处理。这样可以节省相当多的计算时间。

这一结构可以用计算机数据结构理论的树结构表示。树的根对应于整个图像 f，它的叶子对应 f 的像素，每个节点对应 f 的一个正方子域，每个节点有 4 个分枝，从而称为图像的四分树。图像四分树结构的生成示意图如图 8.14 所示。

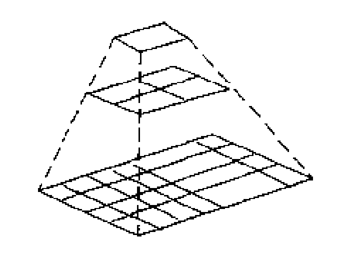


图 8.14　图像四分树结构示意图

8.4.2　利用四分树实现图像分割

利用分开合并算法对图像进行分割的步骤如下：

(1) 生成图像的四分树结构。

(2) 根据经验和任务需要，从四分树的某一层开始，合并满足一致性属性的共根的 4 个子块。重复对图像进行操作，直到不能合并为止。

(3) 考虑上一步中没有合并的子块，如果它的子节点不满足一致性准则，将这个节点永久地分为 4 个子块。如果分出的子块仍不满足一致性准则，继续划分，直到所有的子块都满足为止。

(4) 由于人为地将图像进行四分树分解，可能会将同一区域的象素分在不能按照四分树合并的子块内，因此需要搜索所有的图像块，将邻近的未合并的子块合并为一个区域。

(5) 由于噪声影响或者按照四分树划分区域边缘未对准，进行上述操作后可能仍存在大量的小的区域，为了消除这些影响，可以将它们按照相似性准则归入邻近的大区域内。

区域内和区域间的均一性准则又称“一致性谓词”，可以选择的形式有如下几种：

- 区域中灰度最大值与最小值之差或方差小于某选定值。
- 两区域平均灰度之差及方差小于某选定值。
- 两区域的纹理特征相同。
- 两区域参数统计检验结果相同。
- 两区域的灰度分布函数之差小于某选定值。

另外，还可以根据任务的需要设计其他的区域均一性准则。

MATLAB 提供了对图像进行四分树分解的函数，可以用来实现分开合并算法，下面简单介绍。

1. qtdecomp

qtdecomp 函数实现图像的四分树分解，其语法格式为：

```
S=qtdecomp(I)
S=qtdecomp(I,thereshold)
S=qtdecomp(I,thereshold,mindim)
S=qtdecomp(I,thereshold,[mindim maxdim])
S=qtdecomp(I,fun)
S=qtdecomp(I,fun,p1,p2,…)
```

其中 S=qtdecomp(I)对灰度图像 I 进行四分树分解，返回图像的四分树结构 S。S 为一个稀疏矩阵，若 S(k,m)不等于 0，则象素 I(k,m)对应着分解结构中一个图像子块的左上顶点，图像子块的大小等于 S(k,m)。默认的情况下，S=qtdecomp(I)对图像进行分解，直到各图像块的象素灰度相等。

S=qtdecomp(I,thereshold) 对灰度图像 I 进行四分树分解，直到各图像块的最大灰度与最小灰度之差小于 threshold 为止。

S=qtdecomp(I,thereshold,[mindim maxdim])规定了对灰度图像 I 进行四分树分解各图像子块的最大和最小尺寸。

另外，用 qtdecomp 函数进行四分树分解还可以指定灰度一致性谓词 fun，S=qtdecomp(I,fun)和 S=qtdecomp(I,fun,p1,p2,…)采用函数 fun 来度量图像块内象素的一致性，p1,p2…是

fun 的参数。

例如：

```
I=[ 1    1    1    1    2    3    6    6
    1    1    2    1    4    5    6    8
    1    1    1    1    10   15   7    7
    1    1    1    1    20   25   7    7
    20   22   20   22   1    2    3    4
    20   22   22   20   5    6    7    8
    20   22   20   20   9    10   11   12
    22   22   20   20   13   14   15   16];
S=qtdecomp(I,5);
full(S)
ans =
    4  0  0  0  2  0  2  0
    0  0  0  0  0  0  0  0
    0  0  0  0  1  1  2  0
    0  0  0  0  1  1  0  0
    4  0  0  0  2  0  2  0
    0  0  0  0  0  0  0  0
    0  0  0  0  2  0  2  0
    0  0  0  0  0  0  0  0
```

注意： 因为 S 为稀疏矩阵，因此需要将其转化成完整的矩阵显示出来，才能直观地看到图像四分树分解的结果。S 中非 0 的位置表明了图像块的左上角，非 0 值表示该图像块的边长。

2. qtgetblk

实际上不用把 S 画出也可以获得图像四分树分解的图像块信息。函数 qtgetblk 可以获取四分树分解后子块的象素及其位置信息，其格式为：

```
[val,r,c]=qtgetblk(I,S,dim)
```

它返回图像 I 的四分树分解结构中 dim×dim 大小子块的各个象素的值，存放在 val 矩阵中。同时返回 dim×dim 大小子块的位置，r 为行数，c 为列数。例如对上面例子中的图像 I 进行四分树分解，得到分解结构 S，提取其中大小为 4×4 的图像块，可以采用下面的代码：

```
[vals,r,c]=qtgetblk(I,S,4)
vals(:,:,1)=
    1 1 1 1
    1 1 2 1
    1 1 1 1
    1 1 1 1
vals(:,:,2) =
    20 22 20 22
    20 22 22 20
    20 22 20 20
    22 22 20 20
```

```
r =
     1
     5
c =
     1
     1
```

MATLAB 图像处理工具箱还提供了四分树分解的演示程序 qtdemo，如图 8.15 所示，运行之后可以观察图像分割的结果。

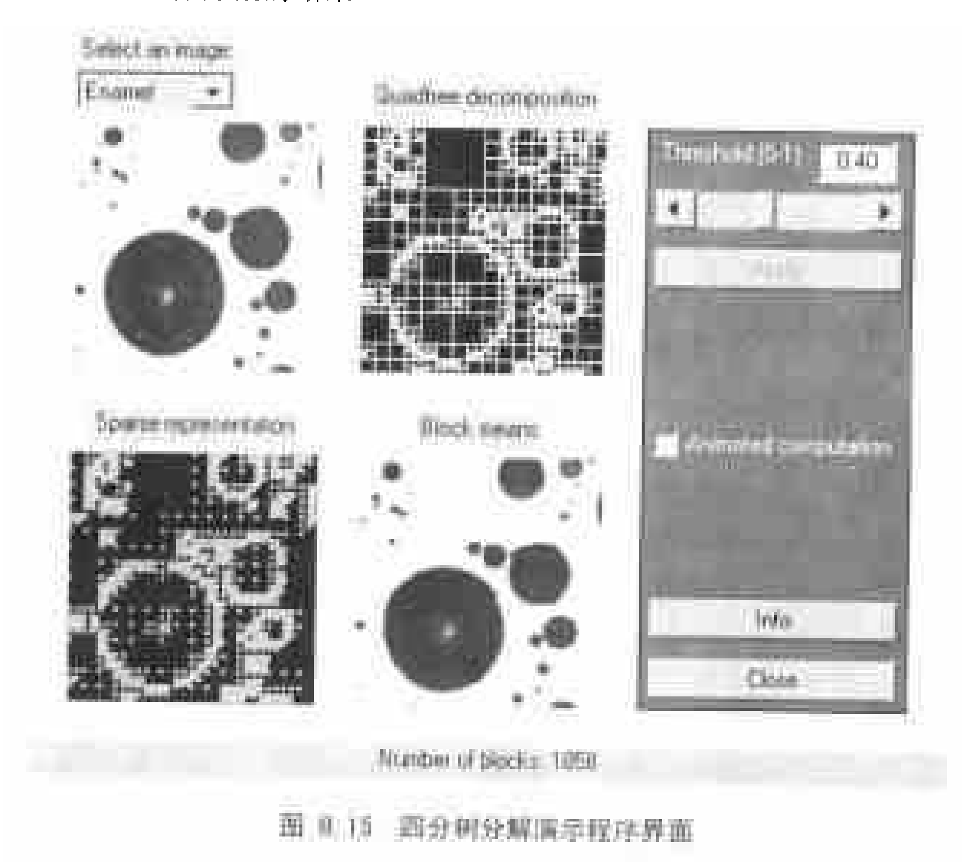


图 8.15 四分树分解演示程序界面

第9章　数学形态学与二值图像操作

近年来，形态学图像处理已发展成为图像处理的一个重要研究领域。关于形态学理论及应用的文章大量出现在各种研究期刊和会议中，许多已经开发和正在开发的应用系统中也使用了数学形态学的理论。目前形态学的应用几乎覆盖了图像处理的所有领域：包括医学图像处理、文字识别、图像编码压缩、材料科学、视觉检测以及计算机视觉等。一些图像分析系统将数学形态运算作为系统的基本运算，并由此出发考虑系统体系结构。形态学方法已经成为从事图像应用领域研究人员的必备工具。

数学形态学处理的对象主要是二值图像。二值图像操作有其特殊的性质，由于二值图像处理相对简单，很多图像处理应用问题都可以转化为二值图像问题，比如医学图像处理中染色体的提取等问题。本章将对二值图像处理的一些方法进行讨论。

9.1　数学形态学图像处理

9.1.1　数学形态学简介

数学形态学图像处理的基本思想是利用一个称作结构元素(structuring element)的“探针”收集图像的信息。当探针在图像中不断移动时，便可考察图像各个部分间的相互关系，从而了解图像各个部分的结构特征。作为探针的结构元素，可直接携带知识（形态、大小、以及灰度和色度信息)来探测所研究图像的结构特点。

数学形态学基于探测的思想与人的视觉特点有类似之处：人总是首先关注一些感兴趣的物体或者结构(比如线结构)，并有意识地寻找图像中的这些结构。这在人的视觉研究中称为注意力集中，简称 FOA(Focus Of Attention)。

从某种特定意义上讲，形态学图像处理是以几何学为基础的。它着重研究图像的几何结构，这种结构表示的可以是分析对象的宏观性质，例如，在分析一个物体的形状时，研究的就是其宏观结构特征；也可以是微观性质，例如，在分析矿石颗粒分布或由小的基元产生的纹理时，研究的是对象的微观结构形态。

通过利用一个结构元素去探测一个图像，看是否能够将这个结构元素很好地填放在图像的内部，同时验证填放结构元素的方法是否有效，可以给出图像的结构信息。在图 9.1 中给出了一个二值图像 A 和一个圆形结构元素 B。这里，A 为需要研究的对象，B 为用于探测的结构元素。很明显，B 可以填放在 A 的左部，但是不能填放在 A 的右部，所以可以

知道图像 A 的左部比结构元素 B 要大，而右部比 B 小，这样就得到了图像 A 的结构特点。通过对图像内适合放入结构元素的位置作标记，便可得到关于图像结构的信息。这些信息与结构元素的尺寸和形状都有关，因而这些信息的性质取决于结构元素的选择。也就是说，结构元素的选择与从图像中抽取何种信息有密切的关系。构造不同的结构元素，便可完成对图像不同性质的分析，得到不同的分析结果。

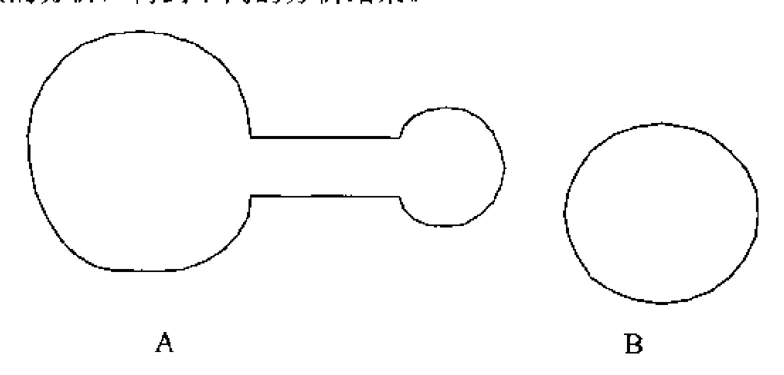


图 9.1 用结构元素B探测图像A

9.1.2 数学形态学的基本运算

数学形态学中最基本的运算是腐蚀和膨胀。腐蚀和膨胀的定义是和集合及集合的运算密切相关的，下面首先介绍数学形态学的基本概念和运算。

设 Ω 为二维欧几里德空间，图像 A 是 Ω 的一个子集，结构元素 B 也是 Ω 的一个子集，$b \in \Omega$ 是欧氏空间的一个点，定义如下两个概念。

- 平移：A^b 定义为图像 A 被 b 平移后的结果，表示为：

$$A^b = \{ a+b \mid a \in A \}$$

  A^b 中所有元素是 A 中的对应元素平移到以 b 为原点的坐标系内的结果。

- 反射：$\tilde{A}$ 定义为图像 A 对于图像原点反射的结果，意即：

$$\tilde{A} = \{ -a \mid a \in A \}$$

图像 A 被 b 平移和被原点反射的结果如图 9.2 所示。

根据上述两个概念，可以定义数学形态学中的两个基本运算：膨胀(dilation)和腐蚀(erosion)。

- 膨胀的运算定义式为：

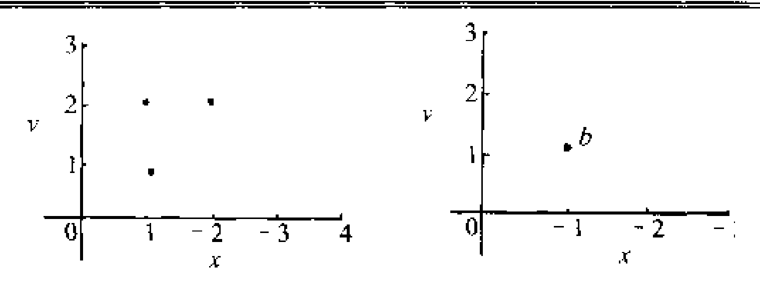
$$A \oplus B = \{ a+b \mid a \in A, b \in B \} = \cup \ A^b$$

- 腐蚀的运算定义式为：

$$A \ominus B = \{ z \in \Omega \mid B^z \subseteq A \} = \cap \ A^b$$

图像 A 被结构元素 B 膨胀，膨胀后 A 形状与结构元素 B 的形状有很大关系。同样，图像 A 被结构元素 B 腐蚀，腐蚀的结果与结构元素 B 的选取有关。因此，选取不同的结构元素 B，同样是做膨胀或腐蚀运算，最后所得的结果可能截然不同。

在定义膨胀和腐蚀运算的基础上，可以定义数学形态学的另外两个常用运算：开运算(opening)和闭运算(clothing)。



定义图像A及点b

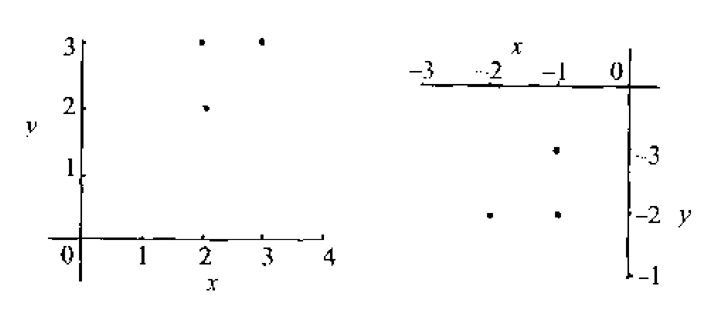


图 9.2　图像A被b平移和被原点反射的结果

- 开运算：A 对 B 的开，即 A 被 B 进行开运算的结果定义为：

$$A \circ B = (A \Theta B) \oplus B$$

即 A 先被 B 腐蚀，再被 B 膨胀的结果。

- 闭运算：A 对 B 的闭运算定义为：

$$A \bullet B = (A \oplus B) \Theta B$$

闭运算的过程与开运算恰好相反，先被 B 膨胀，后其结果再被 B 腐蚀。根据开运算和闭运算的特点，通常可以利用开运算删除图像中的小分支，利用闭运算填补图像中的空穴。

数学形态学的基本运算满足于一些重要的数学条件，此给出最基本的性质：

- 性质 1：膨胀和腐蚀运算具有平移不变性，满足以下条件：

$$(A_p \oplus B) = (A \oplus B)_p$$

$$(A_p \Theta B) = (A \Theta B)_p$$

这个性质说明对图像 A 进行腐蚀和膨胀的运算结果只取决于 A 与 B 的结构，而与 A 的位置无关。

- 性质 2：对开运算和闭运算，满足下列条件：

$$(A \circ B) \subseteq A \subseteq (A \bullet B)$$

这个性质说明开运算可以使图像缩小，闭运算可以使原图像增大。

9.1.3　形态学运算函数

MATLAB 中提供了很多形态学运算函数，能够实现很多形态学操作，下面分别介绍。

1. dilate

函数 dilate 可以实现二值图像的膨胀运算，其语法格式为：

```
BW2=dilate(BW1,SE)
BW2=dilate(BW1,SE,alg)
BW2=dilate(BW1,SE,...,n)
```

其中，BW2=dilate(BW1,SE)：返回图像 BW1 经过结构元素 SE 膨胀后的图像，这里 SE 为结构元素。

BW2=dilate(BW1,SE,alg)：通过参数 alg 设置算法是在空域上实现还是在频域上实现，即

- alg='spatial'时，在空域上实现。
- alg='frequency'时，在频域上实现。

无论在空域上实现还是在频域上实现，运算结果都相同，只是对于大尺寸图像，频域实现较快一些。

BW2=dilate(BW1,SE,...,n)：可以指定对图像进行膨胀运算的次数 n。

对文本图像作膨胀运算，其结果如图 9.3 所示。

```
BW1=imread('text.tif')
SE=ones(6,2)
%定义结构元素。
%SE=   1    1
       1    1
       1    1
       1    1
       1    1
       1    1
BW2=dilate(BW1,SE);
%用结构元素 SE 对 BW1 进行膨胀。
imshow(BW1)
figure,imshow(BW2)
```





图 9.3 对文本图像进行膨胀的结果

2. erode

函数 erode 可以对图像进行腐蚀运算，其语法格式为：

```
BW2=erode(BW1,SE)
BW2=erode(BW1,SE,alg)
BW2=erode(BW1,SE,…,n)
```

其中，BW2=erode(BW1,SE)：返回图像 BW1 经过结构元素 SE 腐蚀后的图像，这里 SE 为结构元素。

BW2=erode(BW1,SE,alg)：通过参数 alg 设置算法是在空域上实现还是在频域上实现，即

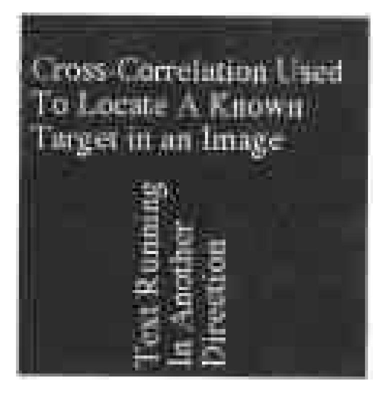
- 当 alg='spatial'时，在空域上实现。
- 当 alg='frequency'时，在频域上实现。

与膨胀运算相同，无论在空域上实现还是在频域上实现，结果相同，只是对于大尺寸图像，在频域上实现要快一些。

BW2=erode(BW1,SE,…,n)：指定腐蚀运算的次数。

对文本图像作腐蚀运算的结果如图 9.4 所示。

```
BW1=imread('text.tif')
SE=ones(6,2)
BW2=erode(BW1,SE);
imshow(BW1)
Figure,imshow(BW2)
```



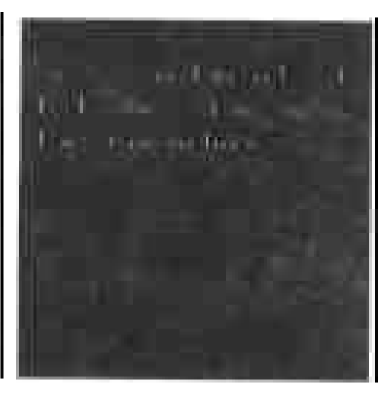


图 9.4　对文本图像进行腐蚀的结果

3. bwmorph

函数 bwmorph 是形态学运算族函数，它的功能和语法格式为：

```
BW2=bwmorph(BW1,operation)
BW2=bwmorph(BW1,operation,n)
```

这里，BW2=bwmorph(BW1,operation)：对图像 BW1 作指定的形态学运算 operation，operation 的可选值及其含义为：

- 'bothat'，闭包运算，即先腐蚀，再膨胀，然后减去原图像。
- 'bridge'，作连接运算，如：

```
1 0 0          1 0 0
1 0 1   变为   1 1 1
0 0 1          0 0 1
```

- 'clean'，去除孤立的亮点，如：

```
0 0 0          0 0 0
0 1 0    变为  0 0 0
0 0 0          0 0 0
```

- 'close'，进行二值闭运算。
- 'diag'，采用对角线填充来去除 8 邻接的背景。
- 'dilate'，用结构元素 ones(3)作膨胀运算。
- 'erode'，用结构元素 ones(3)作腐蚀运算。
- 'fill'，填充孤立的黑点，如：

```
1 1 1          1 1 1
1 0 1    变为  1 1 1
1 1 1          1 1 1
```

- 'hbreak'，断开 H 形连接，如：

```
1 1 1          1 1 1
0 1 0    变为  0 0 0
1 1 1          1 1 1
```

- 'majority'，若像素的 8 邻域中有大于或者等于 5 的元素为 1，则像素值为 1，否则为 0。
- 'open'，执行二值开运算。
- 'remove'，去掉内点，即如果像素的 4 邻域都为 1，则像素值为 0。
- 'shrink'，n=Inf，作收缩运算。这样没有孔的物体收缩为一个点，而含有孔的物体收缩为一个相连的环，环的位置在孔和物体外边缘的中间。收缩运算保持欧拉数不变。
- 'skel'，n＝Inf，提取物体的骨架。即去除物体外边缘的点，但是保持物体不发生断裂，它也保持欧拉数不变。
- 'spur'，去除物体小的分支，例如：
    - thicken，n＝Inf，对物体进行粗化，即对物体的外边缘增加像素，直到原来为连接的物体按照 8 邻接被连接起来。粗化操作也保持欧拉数不变。
    - thin，n=Inf，对物体进行细化，使得没有孔的物体收缩为最小连接棒，而含有孔的物体收缩为一个相连的环，环的位置在孔和物体外边缘的中间。细化运算保持欧拉数不变。
- 'tophat'，用原图像减去开运算后的图像。

例如，用形态学算子去掉图像的内点，抽取图像的骨架，结果如图 9.5 所示。

```
BW1=imread('circles.tif')
Imshow(BW1)
BW2=bwmorph(BW1,'remove');
%用形态学算子去掉图像的内点。
BW3=bwmorph(BW1,'skel',Inf);
%用形态学算子提取图像的骨架。
imshow(BW2)
```

```
figure,imshow(BW3)
```

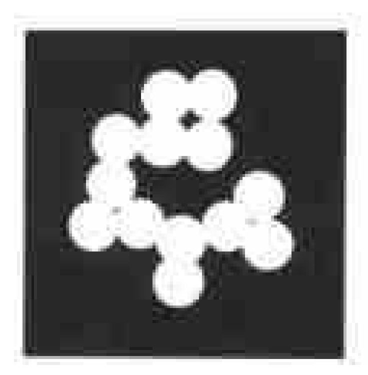



图 9.5 去除图像的内点和抽取图像的骨架的结果

9.2 基于对象的操作

二值图像中，对象是指值为 1 且连接在一起的像素的集合。例如图 9.6 所示的矩阵就代表一个简单的 3×3 正方形对象的二值图像。矩阵中值为 0 的像素代表背景。

```
0 0 0 0 0 0
0 1 1 1 0 0
0 1 1 1 0 0
0 1 1 1 0 0
0 0 0 0 0 0
```

图 9.6 3×3正方形对象的二值图像

9.2.1 四邻域和八邻域

邻接是像素间的基本关系。除图像边缘外，每个像素都有 8 个自然邻点，但是在处理技术中采用两种定义：4 邻接和 8 邻接。4 邻接只认为一个像素的水平和垂直方向上的自然邻点和其邻接，而 8 邻接则认为其 8 个自然邻点都是邻接的。按 4 邻接定义由 4 个邻点组成的邻域称为 4 邻域，按 8 邻接定义的 8 个邻点组成的邻域称为 8 邻域。若两个像点是 4 邻接的，则称它们为 4 连通，若是 8 邻接的，则称它们为 8 连通。

4 邻接和 8 邻接的示意图如图 9.7 所示。

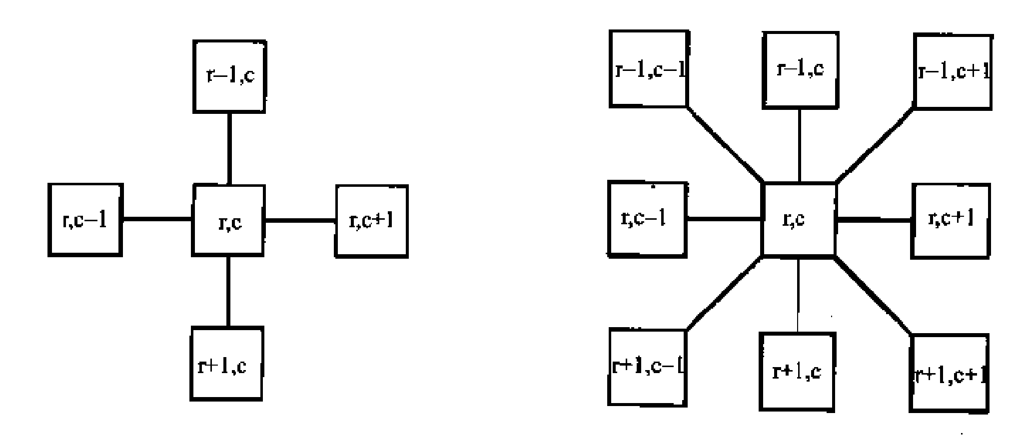


图 9.7 图像四邻接和八邻接示意图

```
0 0 0 0 0 0
0 0 1 1 0 0
0 1 0 0 1 0
0 1 0 0 1 0
0 0 1 1 0 0
0 0 0 0 0 0
```

图 9.8 二值图

由于连通定义不同，同样的一幅数字图像，可能有不同的理解。例如，图 9.8 是一幅二值图，若按 4 连通定义，由 1 所表示的目标是 4 个不连通的直线段，若按 8 连通定义，则是一闭合的环。

9.2.2 边界识别

MATLAB 图像处理工具箱提供的 bwperim 函数用于提取二值图像中对象的边界像素。确定边界时可以指定采用 4 邻接定义或者 8 邻接定义。边界元素必须满足以下两个条件：

- 值为 1。
- 邻域中至少有一个元素值为 0。

bwperim 函数的语法格式为：

```
BW2=bwperim(BW1,n)
```

这里 BW2=bwperim(BW1,n)：返回二值图像 BW1 的边界图像。n 为邻域的类别，n=4 表示采用 4 邻域定义，n=8 代表采用 8 邻域定义，n 的默认值为 8。

下例是用形态学的方法识别图像的边界，结果如图 9.9 所示。

```
BW1=imread('circbw.tif');
BW2=bwperim(BW1,8);
imshow(BW1)
figure,imshow(BW2)
```

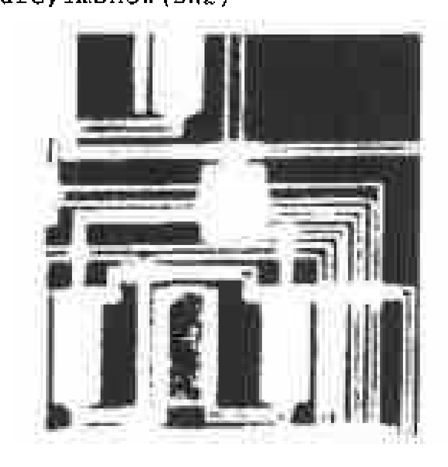
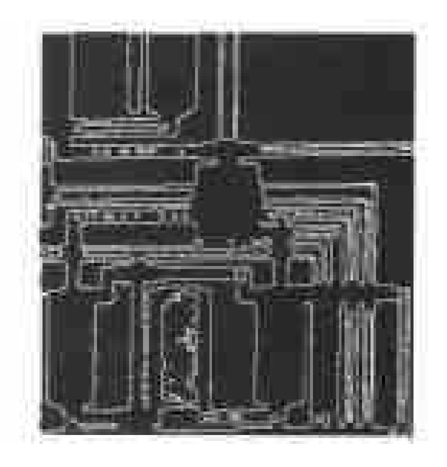



图 9.9 用八邻接探测物体边界

9.2.3 种子填充

种子填充是二值图像处理中的一种常用操作。首先指定一个背景点作为起始点，不断把与之相邻的背景点的值由 0 变为 1，直到到达物体的边界。

种子填充对于去除图像中不规则的形状很有效。比如一幅从照片得到的图像含有一个

圆，本来图像应该显示为一个圆形，但是由于照片反光，图像看起来是一个圆环。在进一步处理之前，可以利用种子填充把圆环恢复成一个圆。

种子填充与其他基于对象操作不同的地方在于它是对背景操作，而不是对前景操作。如果前景是 8 邻接的，则背景是 4 邻接的，反之亦然。

4 邻接和 8 邻接的关系图如图 9.10 所示。

无论前景采用 4 邻接还是 8 邻接，图像都只含有一个物体。但是如果前景采用 8 邻接定义，则物体是封闭的，图像含有两个背景区，即在环内的部分和在环外的部分。如果前景采用 4 邻接定义，物体不是封闭的，图像只含有一个背景区。

```
0  0  0  0  0  0  0  0
0  1  1  1  1  1  0  0
0  1  0  0  0  1  0  0
0  1  0  0  0  1  0  0
0  1  0  0  0  1  0  0
0  1  1  1  1  1  0  0
0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0
```

图 9.10　4邻接和8邻接的关系

MATLAB 中提供的 bwfill 函数可以实现种子填充功能，其语法格式为：

```
BW2=bwfill(BW1,c,r,n)
BW2=bwfill(BW1,n)
[BW2,idx]=bwfill(…)
BW2=bwfill(x,y,BW1,xi,yi,n)
BW2=bwfill(BW1,'holes',n)
[BW2,idx]=bwfill(BW1,'holes',n)
```

其中 BW2=bwfill(BW1,c,r,n)对图像 BW1 进行区域填充，(r(k),c(k))为填充起始点坐标。n 为区域连通数，n＝4 表示采用 4 邻域定义，n＝8 代表采用 8 邻域定义。n 的默认值为 8。BW2=bwfill(BW1,n)由用户交互的指定填充起始点。[BW2,idx]=bwfill(…)返回填充点的线性下标。BW2=bwfill(x,y,BW1,xi,yi,n)允许用户指定特定的坐标系(x,y)，(xi,yi)为相应的填充起始点坐标。BW2=bwfill(BW1,'holes',n)和[BW2,idx]=bwfill(BW1,'holes',n)可以自动搜索并填充图像中的空洞。

下例是对图 9.10 调用 bwfill，并指定 BW1(4,3)为填充起始点，得到

```
BW2＝bwfill(BW1,4,3)
BW2 =
    0    0    0    0    0    0    0    0
    0    1    1    1    1    1    0    0
    0    1    1    1    1    1    0    0
    0    1    1    1    1    1    0    0
    0    1    1    1    1    1    0    0
    0    1    1    1    1    1    0    0
    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0
```

这里采用的是 8 邻接定义。如果前景采用 4 邻接定义，则填充结果充满整个图像，因为如果背景采用 8 邻接定义的话，是一个整体，如下所示。

```
BW2＝bwfill(BW1,4,3,4)
BW2 =
    1    1    1    1    1    1    1    1
    1    1    1    1    1    1    1    1
    1    1    1    1    1    1    1    1
```

```
        1     1     1     1     1     1     1     1
        1     1     1     1     1     1     1     1
        1     1     1     1     1     1     1     1
        1     1     1     1     1     1     1     1
        1     1     1     1     1     1     1     1
```

9.2.4 连通区域标记

MATLAB 图像处理工具箱中提供的 bwlabel 函数可以对二值图像的连接部分进行标记。它主要用于对二值图像中各个分离部分进行标记。用户指定输入图像和特定的边缘约定类型，bwlabel 函数返回与输入图像大小相同的数据矩阵，您可以利用数据矩阵各元素的不同整数值来区分图像中的不同物体。

bwlabel 函数的语法格式为：

```
L=bwlabel(BW,n)
[L,num]=bwlabel(BW,n)
```

这里，L=bwlabel(BW,n)：返回经过标识的图像 L，n 为区域的连通数，n=4 表示采用 4 邻域定义，n=8 代表采用 8 邻域定义，n 的默认值为 8。

[L,num]=bwlabel(BW,n)返回图像 BW 中的连通对象数 num。

假设有二值图像如下：

```
BW1=
      0     0     0     0     0     0     0     0
      0     1     1     0     0     1     1     1
      0     1     1     0     0     0     1     1
      0     1     1     0     0     0     0     0
      0     0     0     1     1     0     0     0
      0     0     0     1     1     0     0     0
      0     0     0     1     1     0     0     0
      0     0     0     0     0     0     0     0
```

调用 bwlabel 函数，同时指定 4 邻接定义：

```
BW2=bwlabel(BW1,4)
```

则返回的数据矩阵为：

```
BW2 =
      0     0     0     0     0     0     0     0
      0     1     1     0     0     3     3     3
      0     1     1     0     0     0     3     3
      0     1     1     0     0     0     0     0
      0     0     0     2     2     0     0     0
      0     0     0     2     2     0     0     0
      0     0     0     2     2     0     0     0
      0     0     0     0     0     0     0     0
```

在 BW2 中，1 代表第 1 个对象，2 代表第 2 个对象，3 代表第 3 个对象。如果采用 8 邻接定义，则只能得到两个对象，即 BW2=bwlabel(BW1,8)，结果如下：

```
BW2 =
     0     0     0     0     0     0     0     0
     0     1     1     0     0     2     2     2
     0     1     1     0     0     0     2     2
     0     1     1     0     0     0     0     0
     0     0     0     1     1     0     0     0
     0     0     0     1     1     0     0     0
     0     0     0     1     1     0     0     0
     0     0     0     0     0     0     0     0
```

注意： bwlabel 输出矩阵不是二值图像，而是 double 类型的矩阵，因此可以用索引色图像的格式显示该图像。方法是先对各元素加 1，使其处于索引色图像正确的范围。这样每个物体显示为不同的颜色，很易于区分，如图 9.11 所示。

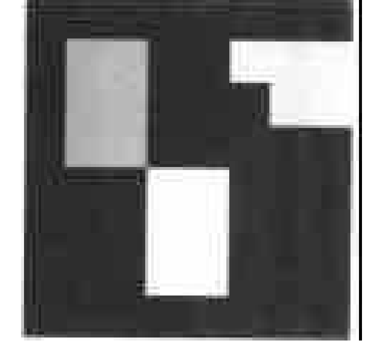


图 9.11　将图像中不同物体表示为不同色彩

例如：

```
X=bwlabel(BW1,4)
Map=[0 0 0;jet(3)]
imshow(X+1,map,'notruesize')
```

9.2.5　选择对象

前面提到，二值图像中，对象是指值为 1，且连接在一起的像素的集合。函数 bwselect 用于在二值图像中选择特定的对象。方法是首先指定一些像素，函数 bwselect 返回在输入图像中包含指定像素的对象的二值图像。

bwselect 函数的语法格式为：

```
BW2=bwselect(BW1,c,r,n)
BW2=bwselect(BW1,n)
[BW2,idx]=bwselect(...)
BW2=bwselect(x,y,BWI,xi,yi,n)
[x,y,BW2,idx,xi,yi]=bwselect(...)
```

这里 BW2=bwselect(BW1,c,r,n)：返回一个包含覆盖在(r,c)处像素的物体的二值图像，r、c 可以是标量或者等长的向量。如果 r、c 是向量，则 BW2 返回任何包含(r(k),c(k))像素的物体。n 为区域的连通数，n=4 表示采用 4 邻域定义，n=8 代表采用 8 邻域定义，n 的默认值为 8。

BW2=bwselect(BW1,n)：用交互的方式确定提取对象的起始坐标。如果忽略 BW1，则在当前的坐标系中操作。单击鼠标左键选定一个起始点，鼠标右键，空格表示删除一个点，按 Enter 键表示结束选择。

[BW2,idx]=bwselect(...)：返回对象点数的线性下标。

BW2=bwselect(x,y,BW1,xi,yi,n)和[x,y,BW2,idx,xi,yi]=bwselect(...)指定非默认的坐标系(x,y)，(xi,yi)为相应坐标系中起始点的坐标。

例如提取文本图像中的字符对象，结果如图 9.12 所示。

```
BW1=imread('text.tif')
C=[16 90 144]
R=[85, 197 247]
BW2=bwselect(BW1,c,r,4)
imshow(BW1)
figure, imshow(BW2)
```



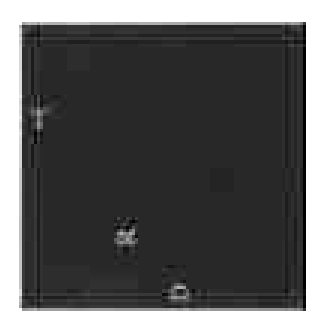


图 9.12 提取文本图像中的字符对象

9.3 特 征 提 取

在进行图像处理时，有时会希望获取图像中改变某些特性的信息。例如在进行膨胀操作时，我们希望知道有多少像素的值由 0 变为 1，或者想知道图像中对象的数目是否改变。利用 MATLAB 图像处理工具箱中的 bwarea 函数和 bweuler 函数可以实现对图像特征的提取操作。

9.3.1 图像面积

bwarea 函数可以获得二值图像的面积，这里的面积简单地可以理解为图像前景中为 1 的像素的个数。

bwarea 函数的语法格式为：

```
total=bwarea(BW)          %返回二值图像 BW 的面积。
```

bwarea 函数并不简单地计算非 0 像素的数目，它对不同的像素赋予不同的权值，以补偿用离散图像代表连续图像的误差。例如一个 50 点长的对角线比 50 点长的水平线长。因此 bwarea 函数返回的 50 点长的水平线面积为 50，而一个 50 点长的对角线长 62.5。

例如，下例计算了电路图(circbw.tif)经过膨胀运算之后图像面积的改变，其示意图如图 9.13 所示。

```
BW1=imread('circbw.tif')
imshow(BW1)
SE=ones(5)
BW2=dilate(BW1,SE)
figure,imshow(BW2)
increase=(bwarea(BW2)-bwarea(BW1))/bwarea(BW1)
increase=
    0.3456
```

由此可见，图像的面积增加了0.3456。

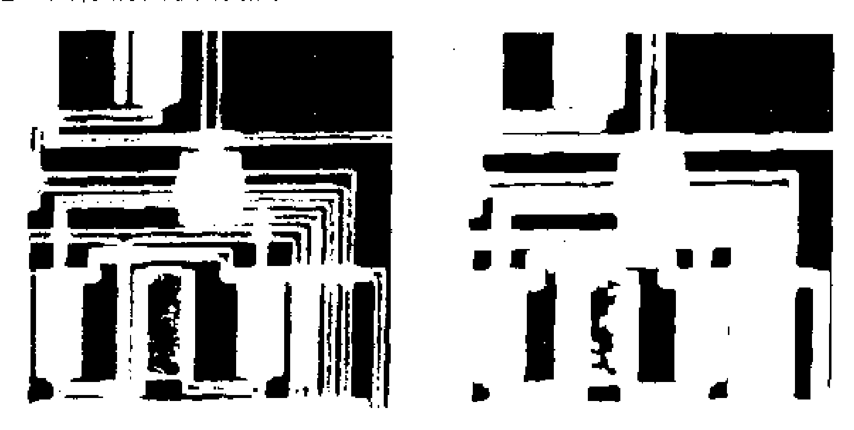


图 9.13 二值图像膨胀运算结果

9.3.2 欧拉数

在几何理论中，闭区域的宏观形态可以用它的拓扑性质来度量。除撕裂或扭接外，在任何变形下都不改变的图像性质称为拓扑性质。显然两点间的距离不是拓扑性质，因为图像拉伸或压缩时它都改变。图像的连通性是拓扑性质，当平移、旋转、拉伸、压缩、扭变之后，连通性是不变的。因此，区域的空洞数H和连通区域数C是拓扑性质。可用欧拉数来度量。

欧拉数是图像的一种拓扑度量。欧拉数等于图像中所有对象的总数减去这些对象中的空洞的数目，即：

```
E=C--H
```

当然，这里的连接也取决于所定义的邻接类型，即4邻接或8邻接。

MATLAB图像处理工具箱中的bweuler函数用来计算二值图像的欧拉数，它的语法格式为：

```
eul==bweuler(BW,n)
```

eul==bweuler(BW,n)：计算二值图像BW的欧拉数，n为连通数，n=4表示采用4邻域定义，n=8代表采用8邻域定义，n的默认值为8。

计算图9.13所示图像的欧拉数，其命令格式如下，最后的结果如图9.14所示。

```
BW=imread('circles.tif')
imshow(BW)
bweuler(BW)
ans =
    -2
```

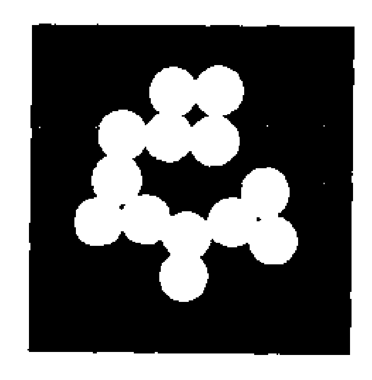


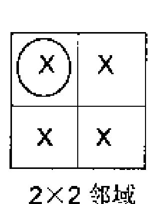
图 9.14 用于计算欧拉数的图像

所以，该图像的欧拉数为-2。

9.4 查 找 表

一些二值图像操作如果采用查找表(Lookup Table)，可能会提高计算的速度。查找表是一个列向量，它把一个像素邻域点的所有可能组合保存起来，使得大量的运算问题转换为查表问题。

函数 makelut 可以用来产生 2×2 和 3×3 邻域的查找表，下面是函数中定义的 2×2 和 3×3 邻域。所有的邻域点都用“x”表示，中心像素用一个圈表示，如图 9.15 所示。



2×2 邻域

X	X	X
X	X	X
X	X	X

3×3 邻域

图 9.15 像素2×2和3×3邻域示意图

对于 2×2 邻域，其元素一共有 16 种排列组合，因此 2×2 邻域的查找表是一个包含 16 个元素的列向量；对于 3×3 邻域，元素一共有 512 种排列组合，它的查找表是 1×512 的列向量。对于超过 3×3 的邻域，例如 4×4 邻域有 65,36 种组合，利用查找表操作就没有太大意义了。

1. makelut

函数 makelut 的语法格式为：

```
lut=makelut(fun,n)
```

它返回函数 fun 定义的查找表，n 为邻域尺寸，n=2 或者 n=3。例如 2×2 邻域的查找表为：

```
f=inline('sum(x(:))>=2')
lut=makelut(f,2)
lut =
     0
     0
     0
     1
     0
     1
     1
     1
     0
     1
     1
     1
     1
```

```
1
1
1
```

2. applylut

函数 applylut 用于基于查找表的二值图像处理，其语法格式为：

```
A=applylut(BW,lut)
```

下面的例子演示如何利用查找表来改变一幅含有文字的图像。首先定义一个函数 fun，使得在 3×3 邻域中，如果有 3 个或者 3 个以上的像素为 1，其他为 0 时，函数 fun 返回 1。然后调用 makelut，把 fun 作为第一个参数传入 makelut，并利用第 2 个参数指定为 3×3 的邻域。

```
f=inline('sum(x(:))>=3')
lut=makelut(fun,3)
```

lut 是一个含有 512 个元素的列向量，每个元素的值是各排列组合在 fun 作用下的结果。由 fun 的定义可知，该查找表具有膨胀的作用。

利用此查找表对一幅图像进行二值操作，结果如图 9.16 所示。

```
BW=imread('text.tif')
A=applylut(BW,lut)
imshow(BW)
figure,imshow(A)
```

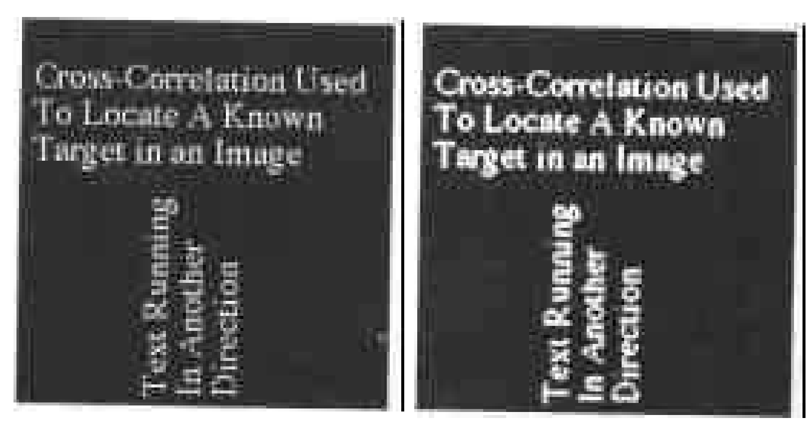


图 9.16　用查找表实现膨胀操作结果

9.5　基于特征的逻辑运算

在二值图像处理中，利用图像的逻辑运算，可以实现基于特征的与运算，以及从图像中提取感兴趣物体的功能。

9.5.1　基于特征的与运算

基于特征的与运算主要是用于以一幅图像为模板，求取另一幅图像与其有重合部分的

对象。下面利用一个例子来说明：

有两幅图像，一幅图像含有 9 个圆，另一个包含一个方框，如图 9.17 所示。

```
load imdemos dots box
imshow(dots)
figure,imshow(box)
```

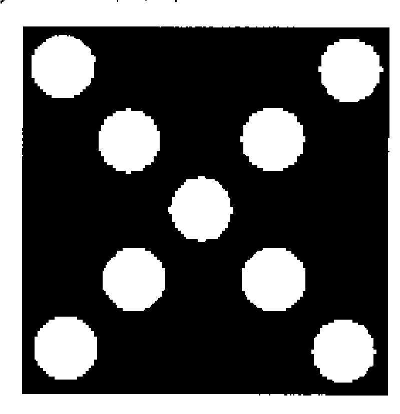
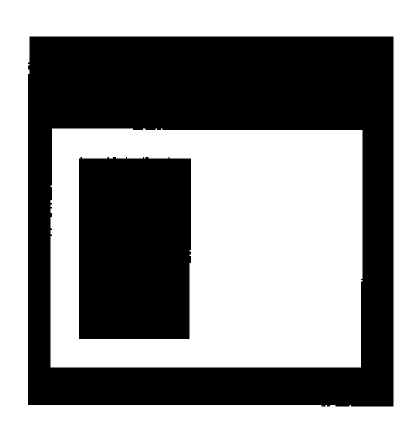



图 9.17 两幅二值图像

首先利用逻辑运算来求取两幅图像中共用的部分，如图 9.18 所示。

```
logical_and=box & dots
imshow(logical_and)
```

然后求出逻辑与中非 0 的部分，将其作为参数传入 bwselect 函数，即可得到基于特征的与运算的结果，如图 9.19 所示。

```
[r,c]=find(logical_and)
feature_and=bwselect(dots,c,r)
imshow(feature_and)
```

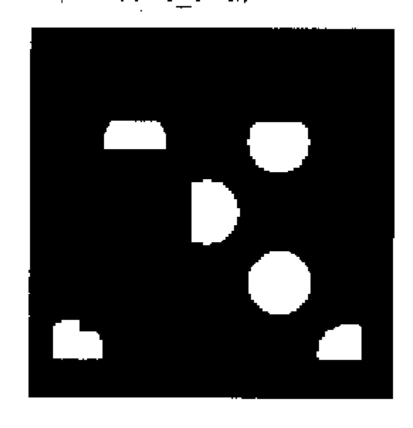


图 9.18 两图像公共部分

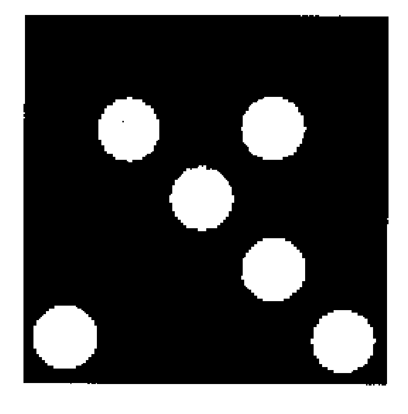


图 9.19 两图像重合部分的物体

从结果可以看出，利用二值图像操作可以用于把两幅图像中有重叠部分的物体提取出来。

9.5.2 利用逻辑运算提取物体

利用二值图像的逻辑运算，还可以从图像中提取感兴趣的物体。这在医学图像处理中

有广泛的应用。下面以从一幅细菌图像中提取细菌数目为例来说明问题。

一幅医学图像，需要从中识别含有亮的细胞核的细菌。我们知道，某些细胞含有一个或多个细胞核，如何从图像中提取含有细胞核的细胞的数目，而不是细胞核的数目，这可以利用二值图像操作来实现。

首先，读入图 9.20 所示的图像。

```
load imdemos bacteria
imshow(bacteria)
```

然后设置阈值，对图像进行分割，并且对分割结果取反，把细胞分割出来。可以看出细胞和细胞核重叠在一起，无法区分，如图 9.21 所示。

```
bact_bw=~(bacteria>=100)
imshow(bact_bw)
```

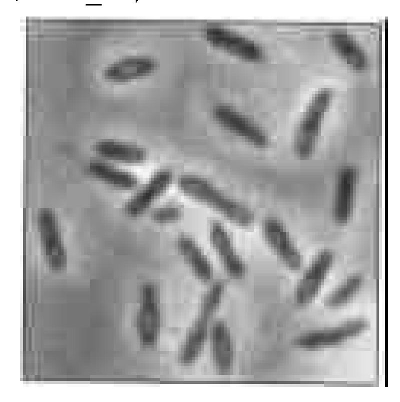


图 9.20　原始医学图像

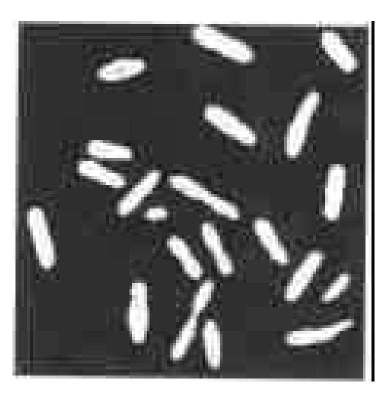
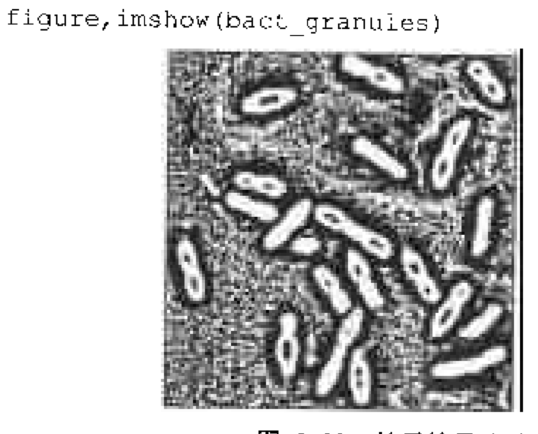


图 9.21　图像灰度分割结果

对原始图像进行 Laplace(拉氏)算子滤波，可突出细胞核与细胞的差异，但是也容易受背景噪声的影响，这可以通过与前面的二值分割掩模进行与运算去除，得到包含在细胞中的细胞核的图像，如图 9.22 所示。

```
filtered=filter2(fspecial('laplacian'),bacteria)
bact_granules=(filtered>-4) & bact_bw;
imshow(filtered)
figure,imshow(bact_granules)
```



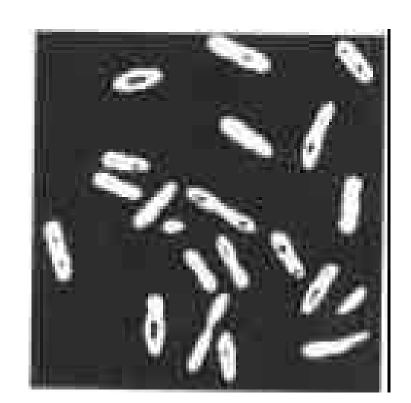


图 9.22　拉氏算子滤波结果及提取的细胞核图像

可以看出细胞核已经从整个细胞中分离出来，但是与背景的值相同，因此需要对二值掩模进行腐蚀运算，并求其与 bact_granules＝0 相同的部分，即可将细胞核孤立出来，如图



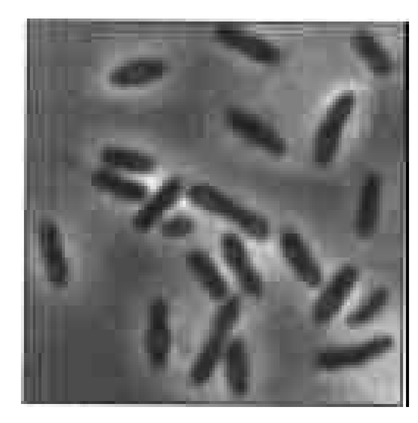
图 9.23 孤立的细胞核

9.23 所示。这里腐蚀运算避免了将细胞边缘的点检测出来。

```
granules=erode(bact_bw)&(bact_granules==0)
imshow(granules)
```

然后利用和 9.5.1 节相似的方法，求出细胞核的坐标，将其传入 bwselect 函数，即可将含有细胞核的细胞提出来，如图 9.24 所示。

```
[r,c]=find(granules)
result=bwselect(bact_bw,c,r)
imshow(bacteria)
figure,imshow(result)
```



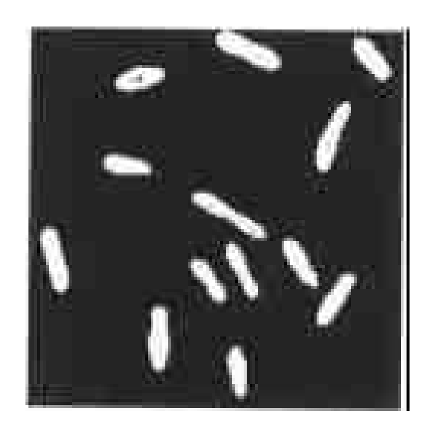


图 9.24 原始图像及含有细胞核的细胞图像

第 10 章　句柄图形与 GUI 设计

在第 4 章我们学习了如何绘制和修饰二维图形和三维图形，这些内容属于 MATLAB 图形系统的高层界面，若想对图形的某些部分做更细致的修饰，就需要掌握 MATLAB 图形系统的低级函数，这就要使用句柄图形(Handle Graphics)。

传统的用户界面是指用户与计算机之间进行交互通信联系的平台。但在最近几年内，这种概念发生了巨大的变化，出现了多种形式的人机交互方式，从命令行的交互方式转变至以图形界面为主的交互形式。现在，图形界面已在人机交互方式中占主导地位，这主要是由于它给用户带来了操作和控制的方便与灵活性。图形用户界面(GUI)是包含图形对象(如窗口、图标、菜单和文本)的用户界面。以某种方式选择或激活这些对象，通常会引起动作或发生变化，最常见的激活方法是用鼠标或其他动作。MATLAB 也提供了在 MATLAB 应用程序中加入 GUI 的功能。

在 MATLAB 系统中，GUI 与图形句柄系统紧密结合在一起，可以说 GUI 是图形句柄系统的子系统。因此本书将二者合在同一章进行讲述，接下来首先讲述句柄图形的概念和使用。

10.1　句 柄 图 形

句柄图形是对底层图形例程集合的总称，它常用来生成图形。这些细节通常隐藏在图形 M 文件的内部，但如果想使用它们也可以得到。句柄图形可以被用来改变 MATLAB 生成图形的方式，不论是只想在一幅图里做一点小变动，还是想影响所有图形输出的方式都可以使用句柄图形。

句柄图形允许定制图形的许多特性，而这用高级命令和前几章中描述的函数则难以实现。例如，如果想用橘黄色来画一条线(plot 命令中可用的颜色不包括这种颜色)，这时我们就可以使用句柄图形来绘制。

本节只对句柄图形概念进行介绍，并提供足够多的信息，使得即使是偶尔使用 MATLAB 的用户也可以利用句柄图形。

10.1.1　图形的对象

句柄图形是基于这样的概念，即一幅图的每一组成部分都是一个对象，每一个对象有一系列句柄和它相关，每一个对象有按需要可以改变的属性。一个对象可以定义为由一组紧密相关、形成唯一整体的数据结构或函数集合。在 MATLAB 中，图形对象是一幅图中

很独特的成份，它可以被单独操作。

由图形命令产生的每一件东西都是图形对象。它们包括图形窗口或仅仅是图形，还有坐标系、线条、曲面、文本和其他。这些对象按父对象和子对象组成层次结构。计算机屏幕是根对象，并且是所有其他对象的父亲。图形窗口是根对象的子对象；坐标系和用户界面对象是图形窗口的子对象；线条、文本、曲面、贴片和图像对象是坐标系对象的子对象。这种层次关系的示意图如图 10.1 所示。

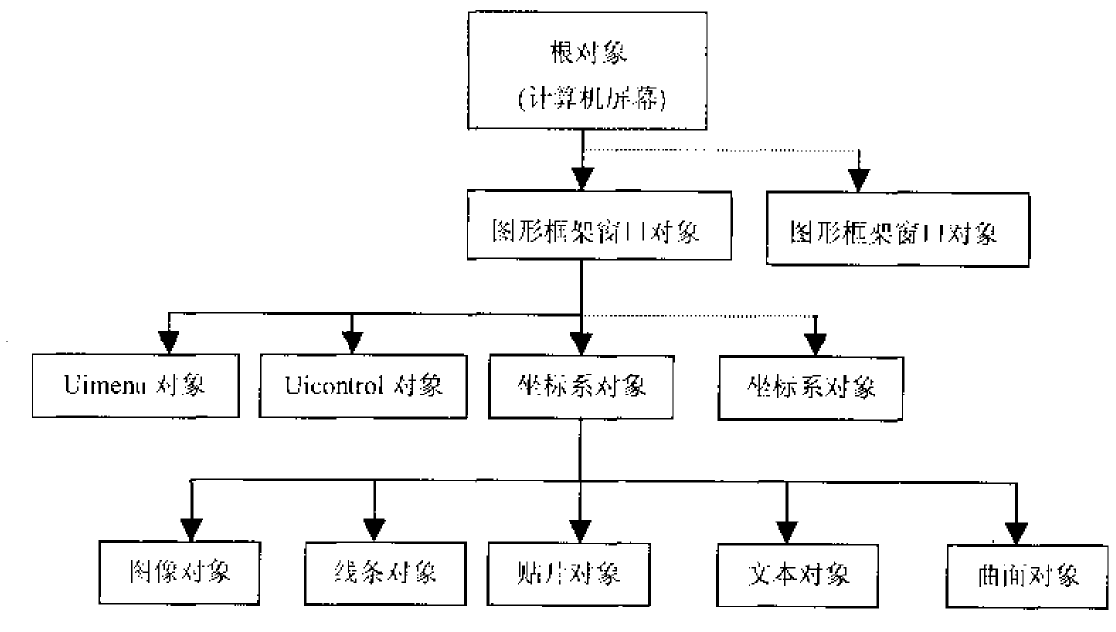


图 10.1 对象层次结构示意图

根可包含一个或多个图形窗口，每一个图形窗口可包含一组或多组坐标轴。所有其他的对象(uicontrol 和 uimenu 外)都是坐标轴的子对象，并且在这些坐标轴上显示。所有创建对象的函数当父对象或对象不存在时，都会创建它们。例如，如果没有图形窗口，plot(rand(size([1:10])))函数会用默认属性创建一个新的图形窗口和一组坐标轴，然后在这组坐标轴内画线。

10.1.2 句柄对象

1. 句柄的概念

假设已打开了 3 个图形窗口，其中两个有两幅子图，如果要改变其中一幅子图坐标轴内一条线的颜色，首先我们应认定想要改变的那条线，在 MATLAB 中，每个对象都有一个数字来标识，叫做句柄(handle)。

每次创建一个对象时，就为它建立一个唯一的句柄。计算机屏幕作为根对象，其句柄常常是零，而其他对象的句柄则是浮点数。

注意，由于精度的要求，最好把浮点数句柄赋值给变量，以备以后使用方便。但绝对不要试图从键盘输入屏幕所见句柄的浮点数。

所有产生对象的 MATLAB 函数都为所建立的每个对象返回一个句柄(或句柄的列向量)。这些函数包括 plot、mesh、surf 及其他。有些图形由一个以上的对象组成，比如，

个网格图由一个曲面组成，它只有一个句柄；而 waterfall 图形由许多线条对象组成，每个线条对象都有各自的句柄。

2. 对象创建函数

在 MATLAB 中除了根对象外，所有的对象都由与之对应的内置函数(Build-in Function)创建，这些函数的功能和调用方式如表 10.1 所示，并且每个函数都返回相应的图形句柄 h。

表 10.1 创建图形对象的底层函数

函数名称	函数功能	调用格式
axes	创建轴对象	Ha_axes=axes('position',[left,bottom,width,height])
figure	创建图形对象	Hf_fig=figure(n)
image	显示图形	Hi_imag=image(X)
line	创建直线对象	Hl_line=line(x,y,z)
patch	填充多边形	Hp_pat=patch(x,y,z,c)
surface	创建曲面对象	Hs_sur=surface(x,y,z,c)
text	创建文字对象	Ht_title=text(x,y,'string')
uicontrol	用户界面控制	Hu_uic=uicontrol('prorerty',value)
uimenu	用户界面菜单	Hu_uim=uimenu('property',value)

每一个底层函数创建的图形对象都被置于适当的父对象中。如果在函数运行之前已经存了相应的父对象，新建的图形对象就会在父对象之中，而且不影响父对象中已经存在的子对象。但是如果适当的父对象不存在，MATLAB 系统就会自动创建所有子对象必需的父对象。

3. 图形对象句柄的获得和删除

正如上面指出的，图形对象句柄可以通过创建过程中读取图形对象底层创建函数的返回值来获得。MATLAB 还提供了以下 3 个函数来获得对象句柄：

gcf：返回当前图形窗口的句柄。

gca：返回当前轴的句柄。

gco：获得当前对象的句柄。

获取图形句柄后，就可以对图形对象进行各种操作，其中包括删除图形对象，其函数为 delete。例如，delete(gca)将删除当前轴和它的所有子对象。

10.1.3 图形对象的属性

所有对象都由属性(property)来定义它们的特征，通过设定这些属性就可以改变图形显示的方式。尽管许多属性是所有的对象都有的，但与每一种对象类型(比如坐标轴、线和曲

面)相关的属性列表都是独一无二的。对象属性可包括诸如对象的位置、颜色、类型、父对象、子对象及其他内容。每一个不同对象都有和它相关的属性，我们可以改变这些属性而不影响同类型的其他对象。

对象属性包括属性名和与它们相关联的值。属性名是字符串,它们通常按混合格式显示,每个词的开头字母大写，比如 LineStyle。但是，MATLAB 识别一个属性时是不分大小写的。另外，只要用足够多的字符来唯一辨识一个属性名即可。例如，坐标轴对象中的位置属性可以用 Position、position，甚至是 pos 来调用。

表 10.2~表 10.9 是 MATLAB 中常用的图形对象的属性列表。当然本书只能介绍图形对象属性中最重要的一小部分。具体的属性列表，用户可以参看 MATLAB 系统的帮助文件。

表 10.2　根对象属性

属性名称	含　义
BlackAndWhite	启动硬件检测标志。{on}为检测显示类型；{off}为显示是单色的，不检测
CurrentFigure	当前图形的句柄
Diary	会话记录。{on}：将所有的键盘输入和大部分输出拷贝到文件中；{off}：不将输入和输出存入文件
DiaryFile	一个包含 diary 文件名的字符串，默认的文件名为 diary
Echo	脚本响应模式。{on}：在文件执行时，显示脚本文件的每一行；{off}：除非指定 echo on，否则不响应
PointerLocation	相对于屏幕左下角指针位置的只读向量[left,bottom]或[x,y]，单位由 Units 属性指定
PointerWindow	含有鼠标指针的图形句柄，如果不在图形窗口内，值为 0
ButtonDowFcn	MATLAB 回调字符串，当对象被选择时传给函数 eval，初始值是一空矩阵
Children	所有图形对象句柄的只读向量
Interruptible	ButtonDowFcn 回调字符串的可中断性。{no}：不能被其他回调中断；{yes}：可以被其他回调中断
Visible	对象可视性。{on}：对根对象有效果；{off}：对根对象无效果

表 10.3　图形对象属性

属性名称	含　义
Color	图形背景色，一个三元素的 RGB 向量或 MATLAB 预定的颜色名，默认的颜色是黑色
Colormap	M×3 的 RGB 向量矩阵，参阅函数 colormap
CurrentAxes	图形的当前坐标轴的句柄
CurrentCharacter	当鼠标指针在图形窗口中，键盘上最新按下的字符键
Currentmenu	最近被选择的菜单项的句柄
CurrentObject	图形内最近被选择的对象的句柄，即由函数 gco 返回的句柄
Currentpoint	一个位置向量[left,bottom]或图形窗口的点的[x,y]，该处是鼠标指针最近一次按下或释放时所在的位置

续表

属性名称	含　义
Keypressfcn	当鼠标指针处在图形内时，按下键，传递给函数 eval 的 MATLAB 回调字符串
Menubar	将 MATLAB 菜单在图形窗口的顶部显示，或在某些系统中在屏幕的顶部显示。{figure}：显示默认的 MATLAB 菜单；{none}不显示默认的 MATLAB 菜单
Mincolormap	颜色表输入项使用的最小数目，它影响系统颜色表。如设置太低，会使未选中的图形以伪彩色显示
Name	图形框架窗口的标题(不是坐标轴的标题)。默认时是空串，如设为 string(字符串)，窗口标题变为"Figure No.n: string"
Nextplot	决定新图作图行为。{new}：画前建立一个新的图形窗口；{add}：当前的图形中加上新对象；{replace}：在画力前，将除位置属性外在的所有图形对象属性重新设置为默认值，并删除所有子对象
Paperposition	代表打印页面上图形位置的向量[left,bottom,width,height]，[left,bottom]代表了相对于打印页面图形左下角的位置，[width,height]是打印图形的大小，单位由 PaperUnits 属性指定
Papersize	向量[width,height]代表了用于打印的纸张大小，单位由 PaperUnits 属性指定，默认的纸张大小为[8.5　11]
paperType	打印图形纸张的类型。当 PaperUnits 设定为归一化坐标时，MATLAB 使用 PaperType 来按比例调整图形的大小。{usletter}：标准的美国信纸；uslegall：标准的美国法定纸张；a3：欧洲 A3 纸；a4letter：欧洲 A4 信纸；a5：欧洲 A5 纸；b4：欧洲 B4 纸；tabloid：标准的美国报纸
Pointer	鼠标指针形状。Crosshair：十字形指针；{arrow}：箭头；watch:钟表指针；topl:指向左上方的箭头；topr：指向右上方的箭头；botl：指向左下方的箭头；botr：指向右下方的箭头；circle：圆；cross：双线十字形；fleur：4 箭头形或指南针形
position	位置向量[left,bottom,width,ehight]，[left,bottom]代表了相对于计算机屏幕的左下角窗口左下角的位置，[width,height]是屏幕大小，单位由 Units 属性指定
Resize	允许不允许交互图形重新定大小。{on}：窗口或以用鼠标来重新定大小；{off}：窗口不能用鼠标来重新定大小
Resizefcn	MATLAB 回调字符串，当窗口鼠标重新定大小时传给函数 eval
Windowbuttondownfcn	当鼠标指针在图形内时，只要按一个鼠标按键，MATLAB 将回调字符串传递给函数 eval
Windowbuttonmotionfcn	当鼠标指针在图形时，只要移动鼠标指针，MATLAB 将回调字符串传递给函数 eval

表 10.4 坐标轴对象属性

属性名称	含　义
Aspectratio	纵横比向量[axis_ratio,data_ratio]这里 axis ratio 是坐标轴对象的纵横比(宽度/高度)，data_ratio 是沿着水平轴和垂直轴的数据单位的长度比。如设置，则 MATLAB 建立一个最大的坐标轴，保留这些比率，该最大轴将在 position 定义的矩形内拟合。该属性的默认值为[NaN,NaN]
Box	坐标轴的边框。{on}：将坐标轴包括在一个框架或立方体内；{off}：不包括坐标轴
Clim	颜色界限向量[cmin　cmax]，它确定将数据映射到颜色映像。cmin 是映射到颜色映像第一个入口项的数据，cmax 是映射到最后一项的数据。参阅函数 cmais
ClimMode	颜色限制模式。{auto}：颜色界限变成轴子对象的数据整个范围；{manual}：颜色界限并不自动改变。设置 Clim 就把 ClimMode 值设为 manual
DrawMode	对象生成次序。{normal}：将对象排序，然后按照当前视图从后向前绘制；fast：按已建立的次序绘制对象，不首先排序
LineStyleOrder	指定线形次序的字符串，用在坐标轴上画多条线例如：'-.|:|--|-'将通过点划线、点线、虚线和实线循环。LineStyleOrder 默认值为 '-' 即只有实线
LineWidth	X,Y 和 Z 坐标轴的宽度，默认值为 0.5
Title	坐标轴标题文本对象的句柄
View	向量[az　el]，它代表了观察者的视角，以度为单位。Az 为方位角或视角相对于负 Y 轴向右转角；el 为 X-Y 平面向上的仰角。
Xcolor	RGB 向量或预定的颜色字符串，它指定 X 轴线、标志、刻度标记和格栅线的颜色。默认为 white(白色)
Xdir	X 值增加的方向。{normal}：X 值从左向右增加；{reverse}：X 值从右向左增加
Xform	一个 4×4 的视力转换矩阵。设置 view 属性影响 Xform
Xgrid	X 轴上的格栅线。On:X 轴上每个刻度标记处画格栅线；{off}：不画格栅线
Xlabel	X 轴标志文本对象的句柄
Xlim	向量[xmin xmax]，指定 X 轴最小值和最大值
Xtick	数据值向量，按此数据值将刻度记画在 X 轴上，将 Xtick 设为空矩阵就撤消刻度标记
XtiVklabels	文本字符串矩阵，用在 X 轴上标出刻度标记。如果是空矩阵，那么 MATLAB 在刻度标记上标出该数字值
Yticklabels	与 X 轴一样
Zticklabels	与 X 轴一样

表 10.5　线条对象属性

属性名称	含　义
LineStyle	线形控制。-：画通过所有数据点的实线；--：画通过所有数据点的虚线；:：画通过所有数据点的点线；-.：画通过所有数据点的点划线；+：用加号作记号，标出所有数据；0：用圆圈作记号，标出所有的数据点；*：用星号作记号，标出所有的数据点；.：用实点作记号，标出所有的数据点；X：用 X 符号作记号，标出所有的数据点
LineWidth	以点为单位的线宽，默认值是 0.5
MarkerSize	以点为单位的记号大小，默认值是 6 点
Xdate	线的 X 轴坐标的向量
Ydate	线的 Y 轴坐标的向量
Zdate	线的 Z 轴坐标的向量

表 10.6　文本对象属性

属性名称	含　义
Extent	文本位置向量[left,bottom,width,height]，[left,bottom]代表了相对坐标轴对象左下角的文本对象左下角的位置，[width,height]是包围文本串的矩形区域的大小，单位由 Units 属性指定
HorizontalAlignment	文本水平对齐。{left}：文本相对于它的 Position 左对齐；{center}：文本相对于它的 Position 中央对齐；{right}：文本相对于它的 Position 右对齐
Position	两元素或三元素向量[XY(Z)]，指出文本对象在三维空间中的位置，单位由 Units 属性指定
Rotation	以旋转度数表示的文本方向。{0}：水平方向；±90：文本旋转±90 度；±180：文本旋转±180°；±270°：文本旋转±270°
String	要显示的文本串
VerticalAlignment	文本垂直对齐。{top}：文本串放在指定的 Y 位置顶部；{cap}：字体的大写字母的高度在指定的 Y 位置；{middle}：文本串放在指定的 Y 位置中央；{baseline}：字体的基线在指定的 Y 位置；{bottom}：文本串放在指定的 Y 位置底部

表 10.7　曲面对象属性

属性名称	含　义
Cdata	指定 Zdata 中每一点颜色的数值矩阵。如果 Cdata 的大小与 Zdata 不同，Cdata 中包含的图像被映射到 Zdata 所定义的曲面
LineWidth	边缘线的宽度，默认值是 0.5 点
MarkerSize	边缘线的记号大小，默认值是非曲直 6 点
Xdata	曲面中点的 X 坐标

续表

属性名称	含　义
Ydata	曲面中点的 Y 坐标
Zdata	曲面中点的 Z 坐标

表 10.8　块对象属性

属性名称	含　义
Cdata	指定沿块边缘每一点颜色的数值矩阵。只有 EdgeColor 或 FaceColor 被设为 interp 或 flat 时才使用
LineWidth	轮廓线的宽度，以点为单位。默认值为 0.5 点
Xdata	沿块边缘点的 X 坐标
Ydata	沿块边缘点的 Y 坐标
Zdata	沿块边缘点的 Z 坐标

表 10.9　图像对象属性

属性名称	含　义
Cdata	指定图像中各元素颜色的值矩阵。Image(c)将 c 赋给 Cdata。Cdata 中的元素是当前颜色映像的下标
Xdata	图像 X 数据：指定图像中行的位置。如忽略，使用 Cdata 中的行下标
Ydata	图像 Y 数据：指定图像中行的位置。如忽略，使用 Cdata 中的行下标

10.1.4　图形对象属性的设置和使用

建立一个对象时，它有一组默认属性值，该值可以用两种方法来改变。一是可以用{属性名，属性值}对来建立对象生成函数；另一个是在对象建立后改变属性。

第一种方法很简单，不再另做具体介绍。值得注意的是使用句柄图形对象创建函数(例如 figure、axis、line 等)接受多个属性名和属性值对。下面的一个小例子可以说明问题。

```
Hf_fig1=figure('color','blue','Numbertitle','off','name','my  figure)
```

说明： 创建的图形窗口，背景为蓝色，标题为 my figure 而不是默认的标题 Figure No.1。

本节着重介绍第 2 种方法。MATLAB 系统中为了获得和改变句柄图形对象的属性，只需要两个函数 get 和 set。

- 函数 get 返回某些对象属性的当前值。使用函数 get 的格式为：

```
get(handle,'PropertyName')      %返回指定对象属性的当前值。
get(handle)     %列出对象的所有属性。
```

举例如下：

```
clf reset;H_mesh=mesh(peaks(20))
```

```
H_grand_parent=get(get(H_mesh,'Parent'),'Parent')
disp('      图柄      轴柄'),disp([gcf gca])    %显示当前图形窗和轴的句柄。
H_mesh=
   73.0007
H_grand_parent=
      1
      图柄      轴柄
1.0      72.0007
```

输出结果如图 10.2 所示。

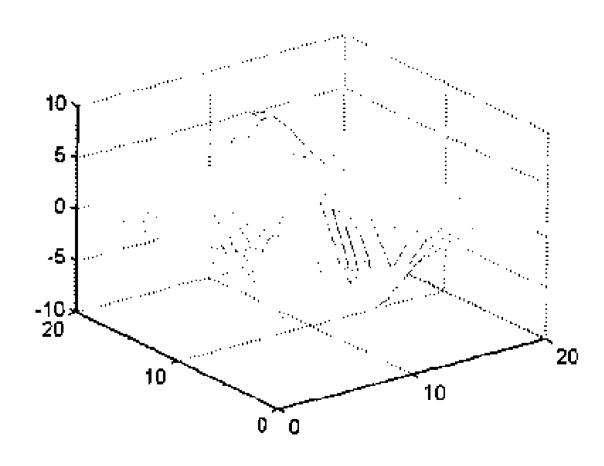


图 10.2 peaks函数网线图

● 使用函数 set 的格式为：

```
set(handle,'PropertyName',value)     %改变句柄图形对象属性。
set(handle,'PropertyName')     %返回一个可赋给对象的属性值列表。
set(handle)     %列出对象所有可设置的属性和可能的取值。
```

一般情况下，函数 set 的第一种调用格式可以有任意的('PropertyName','PropertyValue'对。而在第 2 种格式中如果指定一个没有固定值的属性，那么 MATLAB 系统就会给出提示住处通知用户。

注意： 函数 set 和函数 get 返回不同的属性列表。函数 set 只列出可以用 set 命令改变的属性，这一类属性只可读，但不能被改变，因此它们叫做只读属性、而 get 命令会列出所有对象的属性。

在图 10.2 上，对轴的缺省属性 'DefaultAxesLineStyleOrder' 和 'DefaultAxesColorOrder' 设置所产生的影响：以“红实—蓝实—红虚—蓝虚”的循环次序绘线。

```
clf reset
set(gcf,'DefaultAxesLineStyleOrder','-|:');
set(gcf,'DefaultAxesColorOrder',[1 0 0;0 0 1]);
t=(0:pi/50:2*pi)'; k=0.4:0.1:1;Y=cos(t)*k;
line(t,Y)
```

结果如图 10.3 所示。

接下来是一个较复杂的例子。

```
Hf_fig1=figure     %产生一个具有整数句柄的图形窗口对象。
```

```
Hl_line=line        %产生一个具有浮点句柄的直线对象。
Hf_fig1 =
     1
Hl_line =
   72.0012
```

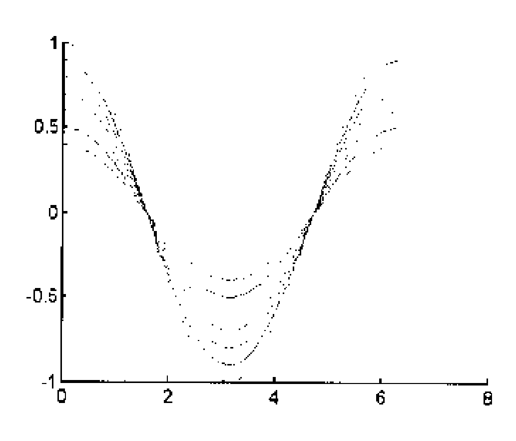


图 10.3 设置轴的默认属性

它要用非标准颜色画一条线。在这里，线的颜色用 RGB 值[1.50]来指不定期，它是适中的橘黄色。

```
x=[-2:0.01:2]*pi;
y=sin(x);                  %产生数据。
Hl_sin=plot(x,y)              %画图并保存图形句柄。
Hl_sin=
72.0013
set(Hl_sin,'color',[1.5 0],'Linewidth',3)%改变线段颜色和线宽。
```

现在加一个紫色的 cos 曲线：

```
z=cos(x);                  %cos 曲线的数据。
hold  on                  %保留 sin 曲线。
Hl_cos=plot(x,z);          %画出 cos 函数曲线并获得句柄。
set(Hl_cos,'color',[0.75  0  1])      %设置颜色为紫色。
hold off
title('sin 和 cos 函数曲线','Fontsize',16,'color','green')
% 给当前图加一个 16 点的绿色标题。
```

输出的结果如图 10.4 所示。

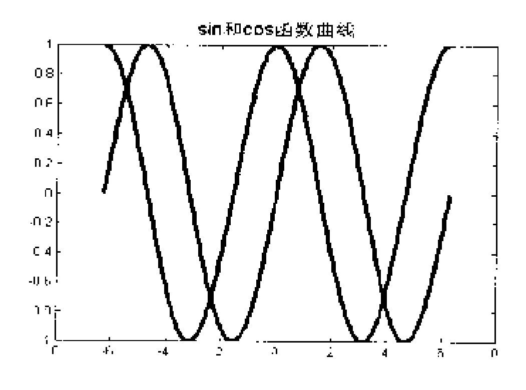


图 10.4 句柄图形对象

10.2　图形用户界面(GUI)设计

MATLAB 的 GUI 的基本图形对象分为两类——用户界面控件对象和用户界面菜单对象，简称为控件对象和菜单对象。设计一个高效的用户界面时，先选择恰当的图形对象，然后将它们有逻辑地组织起来，使得用户界面容易操作使用。

本节将说明图形句柄 uicontrol 和 uimenu 对象的使用，其中把图形界面加到 MATLAB 的函数和 M 文件。uicontrol 对象能建立如按钮、滚动条、弹出式菜单以及文本框等对象，uimenu 对象能在图形窗口中产生下拉式菜单和子菜单。

10.2.1　控件对象及属性

1.　控件对象的类型

控件对象是这样一类图形界面对象，用户用鼠标在控件对象上进行操作，单击鼠标左键时，将会使应用程序作出响应，并执行某些预定的功能子程序(Callback)。控件的操作结果是可见的，有时可以改变应用程序的初始状态。MATLAB 支持 10 种控件对象，它们的添加方式有两种，即基于命令行的和基于 GUI 界面的。下面详细每种控件的目的。

- 坐标轴(Axes)

  坐标轴对象的特性在 9.1 节已经详细讨论过，这里不再详述了。

- 静态文本框(Text)

  静态文本框是仅仅显示一个文本字符串的 uicontrol，该字符串由 String 属性所确定。静态文本框的 Style 属性值是 text，静态文本框一般用于显示标志、用户信息及当前值。

  静态文本框之所以称为“静态”，是因为用户不能动态地修改所显示的文本。文本只能通过改变 String 属性来更改。

- 可编辑文本框(Edit)

  可编辑文本框像静态文本框一样，在屏幕上显示字符。但与静态文本框不同，可编辑文本框允许用户动态地编辑或重新安排文本串，就像使用文本编辑器或文字处理器一样。在 String 属性中即包括该信息。可编辑文本框 Uicontrol 的 Style 属性值是 edit，可编辑文本框一般用来让用户输入文本串或特定值。

  可编辑文本框可包含一行或多行文本。单行可编辑文本框只接受一行输入，而多行可编辑文本框可接受两行以上的输入。单行可编辑文本框的输入以 Enter 键结尾。在 MS-Windows 系统中，多行文本输入以 Ctrl+Return 键结尾，而在 Macintosh 中用 Command+Return 键。

- 弹出式菜单(PopupMenu)

  弹出式菜单一般用于向用户提出互斥的一系列选项清单，让用户可以选择。弹出式菜单不受菜单栏的限制，可位于图形窗口内的任何位置。弹出式菜单的 Style 属性值是 popupmenu。

当关闭时，弹出式菜单以矩形或按钮的形式出现，按钮上含有当前选择的标志，在标志右侧有一个向下的箭头或凸起的小方块来表明 uicontrol 对象是一个弹出式菜单。当指针在弹出式 uicontrol 之上，并单击鼠标左键时，出现其他选项。移动指针可以到不同的选项，松开鼠标左键就关闭弹出式菜单，显示新的选项。MS-Windows 和某些 X Windows 系统平台允许用户单击弹出式菜单，打开它，而后选择其中任一选项来选择。

- 滑标(Slider)

  滑标或称滚动条，包括 3 个独立的部分，分别是滚动槽或长方条区域(代表有效对象值范围)；滚动槽内的指示器(代表滑标当前值)；以及槽的两端的箭头。滑标 uicontrol 的 Style 属性值是 Slider。

  滑标一般用于从几个值域范围中选定一个。滑标值有以下 3 种设定方式：

  - ◆ 鼠标指针指向指示器，移动指示器。拖动鼠标时要按住鼠标按键，当指示器位于期望位置后再松开鼠标按键。
  - ◆ 当指针处于槽中但在指示器的一侧时，单击鼠标按键，指示器按该侧方向移动约等于整个值域范围 10%的距离。
  - ◆ 不论在滑标的哪端单击箭头，指示器都将沿着箭头的方向移动大约为滑标范围 10%的距离。

  滑标通常与所用文本 uicontrol 对象一起显示标志、当前滑标值及值域范围。

- 框架(Frame)

  框架 uicontrol 对象仅是带色彩的矩形区域。框架提供了视觉的分隔性，这点与 uimenu 的 Sepatator 属性相似。框架一般用于组成单选按钮或其他 uicontrol 对象。在其他对象放入框架之前，应事先定义框架，否则框架可能覆盖控制框使它们不可见。

- 命令按钮(PushButton)

  命令按钮又称按钮键或按钮，是小的长方形屏幕对象，常在对象本身上标有文本。将鼠标指针移到对象上来选择命令按钮 uicontrol，单击该按钮，执行由回调字符串所定义的动作。命令按钮的 Style 属性值是 pushbutton。

  命令按钮一般用于执行一个动作，而不是改变状态或设定属性，它与 Windows 系统的命令按钮的作用相同。

- 单选按钮(RadioButton)

  单选按钮又称选择按钮或切换按钮，它由一个标志并和标志文本左端的一个小圆圈或小菱形形成。选择时，圆圈或菱形被填充，且 Value 属必值设为 1；若未被选择，指示符被清除，Value 属性值设为 0。单选按钮的 Style 的属性值是 radiobutton。

  单选按钮一般用于在一组互斥的选项中选择一项。为了确保互斥性，各单选按钮 uicontrol 的回调字符串必须不选组中的其他项，将它们各项的 Value 属性值设为 0。然而，这只是一个约定，如果需要，单选按钮可与复选框交换使用。

- 复选框(CheckBox)

  复选框又称切换按钮，它由具有标志并在标志左边的一个小方框组成。激活时，uicontrol 在检查和清除状态之间切换。在检查状态时，根据平台的不同，方框被

填充或在框内含×，Value 属性值设为 1。若为清除状态，则方框变空，Value 属性值设为 0。

复选框一般用于表明选项的状态或属性。通常复选框是独立的对象，如果需要，复选框可与单选按钮交换使用。

- 列表框(ListBox)

  列表框列出字符串表，用户可以在这个列表中选取单个列表项或多个列表项。对应于选择，MATLAB 执行相应的功能或操作。

2. 控件对象的创建

创建控件对象的方式有两种，下面分别介绍。对于具体的例子将在介绍控件对象属性的修改时一起介绍。

- 基于命令行的方式

  创建控件对象的基本方法是使用 MATLAB 的函数 uicontrol。该函数的调用形式为：

  ```
  h=uicontrol(hfig,属性名，属性值，…)
  ```

  设计程序时，可根据需要向函数 uicontrol 提供必要的属性名和属性值参数对。MATLAB 系统不区别属性字符串中的大小写字母。例如，Style 和 style 都表示是属性 Style 的名字。

  由于控件对象都是图形窗口对象的子对象，所以调用函数 uicontrol 时，第一个参数 hfig 应该是某个图形窗口的句柄值。所创建的控件对象将是该图形窗口的子对象，并出现在该图形窗口中。如果省略这个句柄值参数，则生成的图形对象将是当前图形窗口的子对象。如果此时无图形窗口存在，MATLAB 系统将自动创建一个图形窗口，并在其中创建相应的控件对象。

  h 是所创建控件中对象的句柄值，MATLAB 通过这个句柄值来管理该控件对象，如同 MATLAB 任何类型的图形对象一样。在 MATLAB 图形句柄系统中，控件对象也是一种图形对象，因此对于图形对象的操作方法都能用于控件对象。例如，我们可以用 get 函数取得控件对象的某些属性的当前值，用 set 函数可以改变和重新设置控件对象的某些当前的属性值。

  与控件对象相关联的功能操作是通过控件对象的 Callback 属性来定义的。

- 基于 GUI 的方式

  从 MATLAB 的命令窗口中选择 File | New | GUI 命令，或是在 Command Window 窗口中输入 guide 命令，将出现如图 10.5 所示的窗口。

GUI 设计板是 Guide 工具的集成环境，是 MATLAB 6.1 的新特性之一，它将 MATLAB 5.3 版本的 Guide 控制板工具进行了重组优化。它使得 GUI 图形对象的生成和管理变得更简单直接。例如，我们可以很方便地访问 GUI 图形对象的属性，控制 GUI 图形对象的位置及创建任一种 GUI 图形对象。

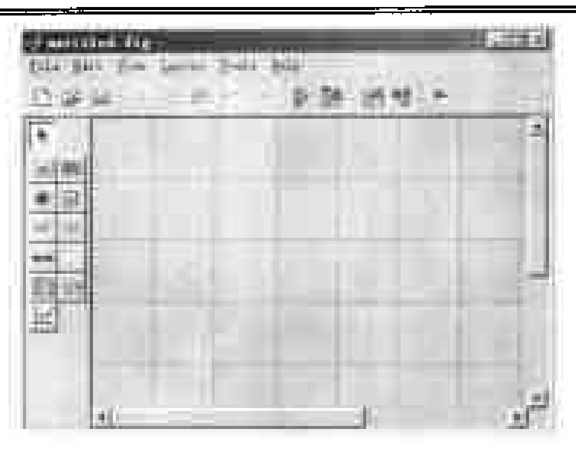


图 10.5 GUI设计板

图 10.5 中有许多工具图标，其中一些图标的使用说明，如表 10.10 所示。

表 10.10 GUI设计板图标按钮

图　标	说　明
(Align Objects)	位置调整器，用于调整图形对象的几何位置
(Edit Menubar)	菜单编辑器，用于编辑菜单
(Property Inspector)	属性浏览编辑器，对应于 MATLAB 5.3 版本中的属性编辑器，用于浏览和编辑 GUI 图形对象的属性
(Objects Browers)	对象浏览器，用于观察各个对象的层次对象
(Activate Figure)	激活图形，用于激活编辑好的 GUI 图形对象
(Select)	选择，以下的按钮用于设计 GUI 的控件对象
(Push Button)	命令按钮
(Radio Button)	单选按钮
(Edit Text)	可编辑文本框
(Slider)	滑动条
(Listbox)	列表框
(Axes)	坐标轴
(Toggle Button)	切换按钮
(Checkbox)	复选框
(Static Text)	静态文本框
(Frame)	框架
(Popup Menu)	弹出式菜单

3. 控件对象的属性

在 MATLAB 的图形句柄系统中，可使用属性对对象的外形和特性进行描述。用户可以在创建控件对象时设定这些属性的属性值，对未指定的属性值，MATLAB 将使用系统提供的默认属性值。用设置图形对象默认属性值的方法可以设置控件对象的默认属性。MATLAB 的控件对象使用相同的属性类型，但是这些属性对于不同类型的控件对象，其含

义不尽相同。

控件对象的属性可分为两大类，第一类是所有的控件对象都具有的公共属性，第 2 类是把控件对象作为图形对象所具有的公共属性。下面分别予以介绍。

- 公共属性
  - Children 属性
    Children 属性的取值为空矩阵，因为控件对象自己没有子对象。
  - Parent 属性
    Parent 属性的取值是某个图形窗口对象的句柄值，因为控件对象总是图形窗口的子对象，该句柄值标明了控件对象所在的图形窗口。如果用 set 函数将 Parent 属性设置为另一个图形窗口的句柄值，则相当于将控件对象移到另一个图形窗口中。
  - Tag 属性
    Tag 属性的取值是字符串。它定义了该控件的一个标识值。定义了 Tag 属性后，在任何程序中都可以通过这个标识值找出该控件对象。
  - Type 属性
    Type 属性的取值总是 uicontrol，这个属性值表明了图形对象的类型。对于控件对象，其类型就是 uicontrol，用户不能改写这个属性。
  - UserData 属性
    UserData 属性的取值是一个矩阵，默认值为空矩阵。用户可以在这个属性中保存与该控件对象相关的重要数据或信息，借此可以达到传递数据或信息的目的。用 set 和 get 函数可以访问该属性。
  - Visible 属性
    Visible 属性的取值是 on(默认值)或 off。如果该属性值为 off，那么控件对象在图形窗口中是不可见的。但是该控件对象仍存在，利用 set 和 get 函数可以访问它的各个属性。
- 基本控制属性
  - BackgroundColor 属性
    BackgroundColor 属性的取值是某些颜色的预定义字符和颜色 RGB 数值，它定义了控件对象区域的背景色，默认值是系统定义的浅灰色。
  - Callback 属性
    Callback 属性的取值是字符串，可以是某个 M 文件名或一小段 MATLAB 语句。当用户激活控件对象(例如，在控件对象图标上单击鼠标左键或移动滑动槽标尺)时，应用程序就运行。Callback 属性定义的子程序(callback 调用)，但是框架静态文本框控件对象不定义 Callback 属性。
  - Enable 属性
    Enable 属性的取值是 on(默认值)、inactive 和 off。当它的取值是 on 时，无论何时激活控件对象 MATLAB，都执行 Callback 属性定义的 callback 子程序。但如果鼠标是在控件对象图标区域外的 5 个像素宽的边界范围内按下时，就执行由 ButtonDownFcn 属性定义的 callback 子程序；若 Enable 属性的取值是

inactive，那么当用户的控件对象图标及 5 个像素边界的范围内单击属性鼠标时，MATLAB 执行 ButtonDownFcn 定义 callback 子程序；当 Enable 属性的取值是 off 时，控件对象的标题字是阴影的，它的 callback 执行功能与 inactive 情形相同，即运行 ButtonDownFcn 定义的 callback 子程序。这样的定义有许多好处，例如可以通过定义 ButtonDownFcn 的 callback 子程序来移动控制对象。

◆ Extent 属性

Extent 属性的取值是 4 元素向量[0,0,width,height]，记录控件对象标题字符的位置大小，度量单位由 Units 属性定义。只能读取该属性值，不能改写。

◆ ForegroundColor 属性

ForegroundColor 属性的取值是某些颜色的预定义字符或颜色的 RGB 数值，这个属性定义控件对象标题字符的颜色。标题字符的默认颜色是黑色。

◆ Max、Min 属性

Max 和 Min 属性的取值都是数值，其默认值分别是 1 和 0。这两个属性值对于不同的控件对象类型意义也不同。以下分别予以介绍：

当单选按钮控件对象被选中时，它的 Value 属性值为 Max 属性的定义的值。当控件对象处于未选中状态时，它的 Value 属性值为 Min 属性的定义的值。

当复选框控件对象被选中时，它的 Value 属性值为 Min 属性定义的值。

当控件对象处于未选状态时，它的 Value 属性值为 Max 属性定义的值。

对于滑标控件对象，Max 属性值必须比 Value 属性值大，Max 定义滑动条最大值，Min 定义滑动条的最小值。

对于可编辑文本框控件对象，如果 Max-Min>1，那么对应的文本框接受多行字符输入；如果 Max-Min<1，那么文本框仅接受单行字符输入。

对于列表框控件对象，如果 Max-Min>1，那么在列表框中允许多行选择，如果 Max−Min<1，那么在列表框中只允许单项选择。

另外，框架、弹出式菜单和静态文本框控件对象不使用 Max 和 Min 属性。

◆ String 属性

String 属性的取值是字符串矩阵或单元数组，它定义控件的标题或选项内容，特别用来定义弹出菜单和列表框之类的多选择的控件对象，它的 String 属性可接受多种形式的输入，如字符串单元数组、字符串矩阵和用“/”分隔的字符串向量。对于 EditBoxes 和 Statictext 这样的多行文字控件对句，String 矩阵的每行、单元数组的每块及字符串的“\n”字符定义的文字行分隔符，即从该处分行。对于 Editable Text 和 String 属性值可以由键盘输入。

◆ Style 属性

Style 属性的取值可以是 Pushbutton(按钮，默认值)、RadioButton(单选按钮)、CheckBoxes(复选框)、Edit Text(文本框)、Statcc Text(静态文本框)、Slider(滚动条)、Frame(框架)、Lstbox(列表框)和 Popupmenu(弹出式菜单)。这个属性值定义控件对象的类型，由相应的值决定。

◆ Units 属性

Units 属性的取值是可以 pixel(像素，默认值)、normalized(相对单位)、inches(英寸)、centimetrers(厘米)、points (磅)。除了 normailzed(相对单位)以外，其他都是绝对单位。这个属性值作为解释 Extend 和 Position 属性值的度量单位。所有单位的度量都是从图形窗口的左下角处开始。在相对单位下，图形窗口的左下角对应(1.0,1.0)。

◆ Value 属性

Value 属性的取值可以是向量值，也可以是数值。它的含义以及解释依赖于控件对象的类型。对应于单选按钮和复选框对象，当它们处于选中状态时，Value 属性值由 Max 属性值定义，反之由 Min 属性值定义。对于滑动条对象，Value 属性值处于 Min 和 Max 属性值之间，由滑动条标尺位置对应的值定义。对于弹出式菜单对象，Value 属性值是被选项的序号，例如 1 对应第一项。所以由 Value 值，可知弹出式菜单的选项。同样，对于列表框对象，Value 属性值定义了列表框中亮度选项的序号。其他的控件对象不使用这个属性值。

- 修饰控制属性

◆ FontAngle 属性

FontAngle 属性的取值是 normal(默认值)、italic 和 oblique，这个属性值定义控件对象标题等的字体。其值为 normal 时，选用系统默认的正字体；其值为 italic 或 oblique 时，使用方头斜字体。

◆ FontName 属性

FontName 属性的取值是控件对象标题等使用字体的字库名，必须是系统支持的各字库。默认字库是系统的默认字库。

◆ FontSize 属性

FontSize 属性的取值是数值，它定义控件对象标题等字体的字号。字号单位由 FontUnits 属性值定义，默认值与系统有关。

◆ FontUnits 属性

FontUnits 属性的取值是 points(磅，默认值)、normalized(相对单位)、inches(英寸)、centimeters(厘米)或 pixels(像素)，该属性定义字号单位。相对单位将 FontSize 属性值解释控件对象图标高度百分比，其他单位都是绝对单位。

◆ FontWeight 属性

FontWeight 属性的取值是 normal(默认值)、light、demi 或 bold，它定义字符字体的粗细。

◆ HorizontalAlignment 属性

HorizontalAlignment 属性的取值是 left、center(默认值)或 right，它定义控件对象标题等的调整方向，即标题在控件对象图标上居左(left)、居中(center)或居右(right)。

- 辅助属性

◆ ListboxTop 属性

ListboxTop 属性的取值是数量值，这个属性只用于 Listbox 控件对象，它定义了位于列表框中最上方的字符串在 String 属性的序号。非整数值被截断为大

于该值的最小整数值。

◆ SliderStep 属性

SliderStep 属性的取值是二元素向量[minstep,maxstep]，这个属性只对 Sliders 控件对象的定义。Maxstep 定义了在滑动槽内单击鼠标左键时，滑动槽标尺应该移动的距离相对于滑槽长度的百分比；minstep 定义了滑动槽两端箭头上单击鼠标左键时，滑动槽标尺应该移动的距离相对于滑动槽长度的百分比。它的默认值是[0.01 0.10]。

◆ Selected 属性

Selected 属性的取值是 on 或 off(默认值)。当 Selected 属性值是 on 时，如果 SelectedHighligh 属性值也是 on，那么 MATLAB 就显示一个对象被选中的标记。如果将 ButtonDownFcn 定义为设置这个属性为 on，那么当选中控件对象时，控件对象就会有一个被选中的方框。

◆ SelectionHighligh 属性

SelectionHighligh 属性的取值是 on(默认值)或 off。该属性值是 on 时，该属性决定控件对象被选中时，是否显示被选中的状态。这个属性必须与 Selected 属性配合使用，共同控制控件对象被选中时的状态。

● Callback 管理属性

◆ BusyAction 属性

BusyAction 属性的取值是 cancel 或 pueue(默认值)，该属性决定 MATLAB 采取的控制中断执行控件对象的 Callback 调用的方式。在 MATLAB 环境中，如果有一个 Callback 调用在运行，那么随后的 Callback 调用总是在试图中断正在运行的 Callback 调用；如果 Interruptible 的属性值为 off,那么 BusyAction 属性就决定 MATLAB 处理事件的方式。此时，如果它的值是 cancel，就从事件队列中取消企图运行第 2 个 Callback 调用的事件加入事件队列，直至当前的 Callback 调用成功为止。

◆ ButtonDownFcn 属性

ButtonDownFcn 的取值是字符串，一般是某个 M 文件名或一小段 MATLAB 语句。当用户在围绕控件对象的 5 个像素宽的边界内按下鼠标左键时，程序执行该属性定义的 Callback 调用。控件一般是矩形区域。如果控件对象的 Enable 属性值是 inactive 或 off，则当在该区域以及 5 个像素宽的边界内按下鼠标左键时，就执行 ButtonDownFcn 属性定义的 Callback。如果这个 Callback 定义为表达式，那么计算结果就属于 MATLAB 工作区。

◆ CreateFcn 属性

CreateFcn 属性的取值是字符串，一般是某个 M 文件名或一小段 MATLAB 语句，即当 MATLAB 生成控件对象时，MATLAB 事先应该执行的程序段。这个属性只能按设置默认值的方式定义，例如：

```
Set(0,'DefaultUicontrolCreateFcn', set(gcf,'IntegerHandle','off")')
```

从根对象层设置控件对象的默认值。对于已经存在的控件对象，改变该属性值，对该对象无任何影响。MATLAB 在生成控件对象并完成所有默认属性值

的设置后，再执行由 CreateFcn 定义的 Callback。

如果一个控件对象 CreateFcn 属性定义的 Callback 正在运行，那么该控件对象的句柄值不可访问，必须用函数 gcbo 取得。

- DeleteFcn 属性

  DeleteFcn 属性的取值是字符串，一般是某个 M 文件名或一小段 MATLAB 语句。当删除控件对象时，MATLAB 在清除该对象属性之前，要执行由 DeleteFcn 属性定义的 Callback。如果一个控件对象 DeleteFcn 属性定义的 Callback 正在运行，那么该控件对象的句柄值不可访问，必须用函数 gcbo 取得。

- HandleVisibility 属性

  HandleVisibility 属性的取值为 on(默认值)、off 或 callback，这个属性定义控件对象句柄的可访问权限，决定其句柄值是否在对象的 Children 属性中出现。如果 HandleVisibility 属性值是 on，那么它的句柄值总是可见的。如果属性值为 callback，那么该句柄值只在该对象的 Callback 调用，以及 Callback 调用的函数内是可见的，但是命令行内调用的函数不能访问这样的控件句柄。也就是说，这样的句柄对于控件对象自己的 Callback 调用是私有的。这对于保护交互式的 HandleVisibility 属性值被设置为 off，那么控件对象的名柄在任何时候都是不可见的。如果句柄是不可见的，那么任何查询句柄的函数都不能返回它的值，这些函数包括 get、findobj、gcf、gco、newplot、cla、clf 和 close 等。当 HandleVisibility 属性值是 callbacka 或 off 时，这样的控件对象不会出现在图形窗口对象的 Childrent 和 CurrentObject 属性中，也不会出现现在根对象 CallbackObject 属性中。但是可以将根对象的 ShowHiddenHandles 属性设置为 on，临时使所有的句柄都是可见的，而不管其 HandleVisibility 属性值是什么。被隐藏的句柄仍然是有效的，只要知道这样的句柄值，一切关于句柄的操作都是合法的。

- Interruptible 属性

  Interruptible 属性的取值是 on 或 off(默认值)，这个属性决定控件对象的 Callback 是否可以被随后的 Callback 调用中断。只有由 ButtonDownFcn 和 Callback 属性定义的 Callback 受 Interruptible 属性值的影响。在运行程序时，只有遇到 drawnow、figure、getframe 和 pause 等命令，MATLAB 系统才检查可以中断程序运行的 Callback 调用事件。

4. 控件对象属性的修改

控件对象属性的创建和修改有两种方式——传统的基于命令行的方式和使用 GUI 界面的方式。这两种方式的本质相同，都是调用 set 函数对句柄进行操作。但是，在编制程序界面时，GUI 界面显得比较简单好用；而在运行程序中，如果要创建新的对象、改变已有的对象属性，就只能使用 uicontrol 和 set 语句了。因而这两种方法都很重要，而 set 函数的使用则是基础。接下来将通过具体例子进行介绍。

- 使用命令行创建控件对象及处理对象的属性

  使用这种方法，可以在程序开始时设置控件对象的属性，也可在程序运行中设置。

在程序开始时，系统对控件对象的属性给出默认值，但是往往不能满足要求，因而需要用户进行自定义。对此，可以使用函数 **uicontrol**，它的语法是：

```
H=uicontrol('propertyNanmel',valuel,'propertyName2',value2,…)
H 是该函数的返回顾值，为所创建对象的句柄。
```

Uicontrol 函数的具体用法前面已经介绍过，这里就不再详述了。

要创建种控件对象，只要如前所述，将它的 Style 指定为所需的值就可以了。为了方便查阅，列出表 10.11。

表 10.11　Style属性取值表

属性取值	所创建的对象
PushButton	命令按钮
ToggleButton	切换按钮
RadioButton	单选按钮
CheckBox	复选框
Edit	可编辑文本框
Static Text	静态文本框
Slider	滑动条
Frame	框架
Listbox	列表框
PopuPmenu	弹出式菜单

在 MATLAB 命令窗口中输入下面命令可以创建一个按钮对象：

```
Hbutton=uicontrol('style','pushbutton');
```

对以上命令的说明如下：

- uicontrol 函数使用系统的默认值(Factory Value)作为默认值，但是系统默认值是可以修改的。因此在需要创建大量对象时，可以考虑修改默认值(方法见后)，进而创建出大量符合自己要求的对象。
- 通过返回的句柄值，可以对图形对象进行操作。

下面是一个综合例子，通过它，用户可以了解和熟悉 uicontrol 的用法。

对于传递函数为 $G=\dfrac{1}{s^2+2\zeta s+1}$ 的归一化二阶系统，制作一个能绘制该系统单位阶跃响应的图形用户界面。本例演示图形界面的大致生成过程、静态文本和编辑框的生成、坐标方格控制键的形成及如何使用该界面。具体步骤如下：

(1) 产生图形窗口和坐标轴框，结果如图 10.6 所示。

```
clf reset
H=axes('unit','normalized','position',[0,0,1,1],'visible','off');
set(gcf,'currentaxes',H);
str='\fontname{隶书}归一化二阶系统的阶跃响应曲线';
text(0.12,0.93,str,'fontsize',13);
h_fig=get(H,'parent');
```

```
set(h_fig,'unit','normalized','position',[0.1,0.2,0.7,0.4]);
h_axes=axes('parent',h_fig,...
   'unit','normalized','position',[0.1,0.15,0.55,0.7],...
   'xlim',[0 15],'ylim',[0 1.8],'fontsize',8);
```



图 10.6　产生坐标轴

提示：　“...”在这里是表示换行，因而可以将每个属性的设置单独写在一行之中，这样有利于增强程序的可读性。

(2) 在坐标框右侧生成作解释用的“静态文本”和可接受输入的“编辑框”，结果如图 10.7 所示。

```
h_text=uicontrol(h_fig,'style','text',...     % 创建静态文本框。
   'unit','normalized','position',[0.67,0.73,0.25,0.14],...
   'horizontal','left','string',{'输入阻尼比系数','zeta ='});
h_edit=uicontrol(h_fig,'style','edit',...     % 创建可编辑文本框。
   'unit','normalized','position',[0.67,0.59,0.25,0.14],...
   'horizontal','left',...
   'callback',[...
      'z=str2num(get(gcbo,''string''));',...
      't=0:0.1:15;',...
      'for k=1:length(z);',...
      's2=tf(1,[1 2*z(k) 1]); ',...
      'y(:,k)=step(s2,t);',...
      'plot(t,y(:,k));',...
      'if (length(z)>1) ,hold on,end,',...
      'end;',...
       'hold off,']);
```



图 10.7 在图形界面中添加编辑框和文本框

(3) 形成坐标方格控制按键，结果如图 10.8 所示。

```
h_push1=uicontrol(h_fig,'style','push',...    %创建命令按钮。
   'unit','normalized','position',[0.67,0.37,0.12,0.15],...
   'string','grid on','callback','grid on');
```

```
h_push2=uicontrol(h_fig,'style','push',...
   'unit','normalized','position',[0.67,0.15,0.12,0.15],...
   'string','grid off','callback','grid off');
```



图 10.8 添加了两个按键的图形界面

(4) 输入阻尼比系数ζ，可得单位阶跃响应曲线，如图 10.9 所示。单击 grid on 按钮，产生网格，结果如图 10.10 所示。

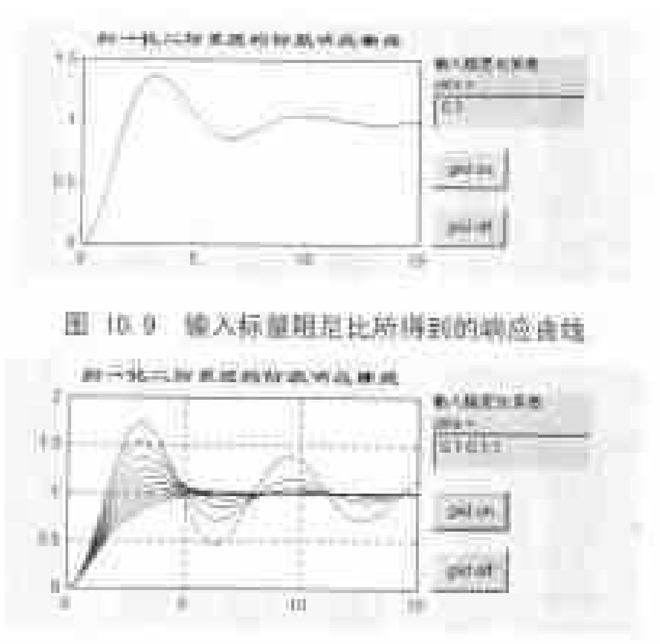


图 10.9 输入标量阻尼比所得到的响应曲线

图 10.10 输入阻尼比数组所得到的一组响应曲线

● 使用 GUI 设计板创建控件对象及处理对象的属性

MATLAB 提供了一个 GUI 界面的工具，可以用来创建改控件对象，也可以修改控件对象的属性。通过上面同一个例子来简要说明 GUI 设计板的使用方法。

如前所述，从 MATLAB 的命令窗口中选择 File | New/GUI 命令，或是在 Command Window 窗口中输入 guide 命令，出现图 10.5 所示的 GUI 设计板窗口，该窗口默认为一个图形对象。

(1) 产生图形窗口和坐标轴框。

单击设计板窗口中左侧的坐标轴 (Axes)图标按钮，在窗口中间画出一个坐标轴，如图 10.11 所示，这样就创建了一个坐标轴对象，该对象是默认图形对象的子对象。我们可以对控件对象的属性进行修改，为此先选中坐标轴对象，再单击窗口中右上端的 Property Inspector (属性浏览编辑器)图标按钮，弹出如图 10.12 所示的属性浏览编辑器，窗口左端为坐标轴对象的属性名称，右端为属性值。作如下属

性设置：

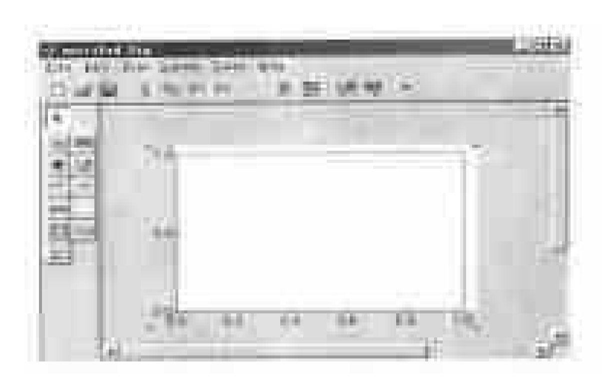


图 10.11　创建一个坐标轴对象

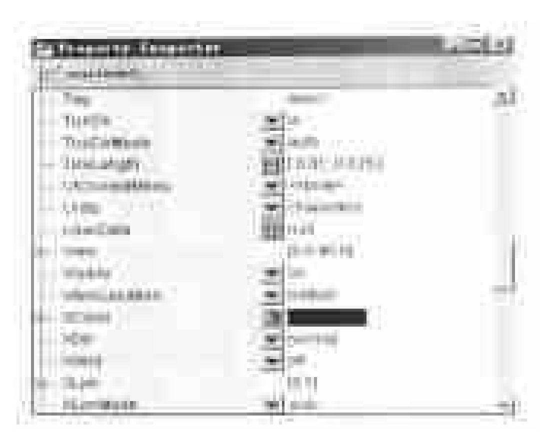


图 10.12　属性浏览编辑器

```
'unit','normalized',
'position',[0.1,0.15,0.55,0.7],
    'xlim',[0 15],
'ylim',[0 1.8],'fontsize',8)
```

单击设计板窗口中左侧的 Static Text (静态文本框)图标按钮，在坐标轴上方画出一个静态文本框，同理可用属性浏览编辑器作如下属性设置：

```
'fontname','隶书',
'string','归一化二阶系统的阶跃响应曲线',
'posision.x', 0.12,
'posision.y', 0.93,
'fontsize',13
```

单击设计板窗口中右上端的 Activate Figure (激活图形)按钮，则生成如图 10.7 所示的图形。

(2) 在坐标框右侧生成作解释用的静态文本和可接受输入的编辑框。

单击设计板窗口中左侧的 Static Text (静态文本框)图标按钮，在坐标轴右侧画出一个静态文本框，用属性浏览编辑器作如下属性设置：

```
'unit','normalized',
'position',[0.67,0.73,0.25,0.14],
'horizontal','left',
'string',{'输入阻尼比系数','zeta ='}
```

单击设计板窗口中左侧的 Edit Text (可编辑文本框)图标按钮，在坐标轴右侧画出一个可编辑文本框，用属性浏览编辑器作如下属性设置：

```
'unit','normalized',
'position',[0.67,0.59,0.25,0.14],
'horizontal','left',...
'callback',[...
'z=str2num(get(gcbo,''string''));',...
't=0:0.1:15;',...
'for k=1:length(z);',...
's2=tf(1,[1 2*z(k) 1]); ',...
'y(:,k)=step(s2,t);',...
```

```
'plot(t,y(:,k));',...
'if (length(z)>1) ,hold on,end,',...
'end;',...
'hold off,']);
```

单击设计板窗口右上端的 Activate Figure (激活图形)按钮，则生成如图 10.8 所示的图形。

(3) 形成坐标方格控制按键。

单击设计板窗口中左侧的 Push Button (命令按钮)图标按钮，在坐标轴右侧画出一个命令按钮，用属性浏览编辑器作如下属性设置：

```
'unit','normalized',
'position',[0.67,0.37,0.12,0.15],
'string','grid on',
'callback','grid on');
```

单击设计板窗口中左侧的 Push Botton 图标按钮，在坐标轴右侧再画一个命令按钮，用属性浏览编辑器作如下属性设置：

```
'unit','normalized',
'position',[0.67,0.15,0.12,0.15],...
'string','grid off',
'callback','grid off');
```

单击设计板窗口右上端的 Activate Figure (激活图形)按钮，则生成如图 10.8 所示的图形。

(4) 输入阻尼比系数ζ，按 Enter 键可得单位阶跃响应曲线，单击 grid on 按钮，产生网格。

提示：

- 对于控件对象位置的调整，可以直接用鼠标拖动对象到适当位置。
- 单击设计板窗口右上端的 Objects Brower(对象浏览器)按钮，可以浏览上例中创建的各对象层次关系，如图 10.13 所示。

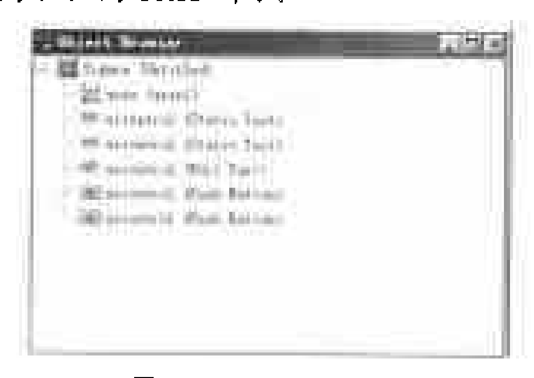


图 10.13 对象浏览器

10.2.2 菜单对象及属性

菜单对象(或称下拉式菜单对象)可以让用户在运行应用程序时，从一批功能选择项中

浏览和选择某项功能。在 MATLAB 的每个图形窗口上，有一个主菜单栏，所有的主要菜单都列在菜单栏上。菜单的标题或名字简单地描述了该菜单的功能。

在 Windows 系统中，菜单栏在图形窗口的标题栏下面。File、Edit、Windows 和 Help 菜单是 MATLAB 图形窗口中的默认主菜单。如果要取消图形窗口的默认的主菜单，可以将图形窗口的 MenuBar 属性事先设置为 none，然后再创建用户自定义的主菜单。

单击主菜单栏上的主菜单时，菜单会自动下拉式打开，显示出该主菜单对象的命令或它的子菜单。选择一个命令就意味着让应用程序执行某种预定义的功能和操作。例如，在 File 主菜单中有一个命令 Print，当用户选择该命令时，即可以打印该图形窗口中的图形。

1. 菜单对象的创建

与上面所讨论过的控件对象一样，菜单对象的创建也有两种方式——基于命令行的编程方式和基于 GUI 的方式。下面分别介绍这两种方法。

- 基于命令行的方式

  菜单对象的创建函数是 uimenu，可以用于创建菜单对象和命令对象，但其调用方式不同。这里所说的对象，简称菜单面，是指本身不再具有子菜单的功能选项，只对应于某种功能操作。而菜单对象或称子菜单项是指自身包含有下一级命令，它的功能就是打开它的子项。两者的区别在于，命令对象的 Children 属性值为空矩阵，而菜单对象的 Children 属性值为图开窗口句柄值或另一个菜单对象的句柄值。

  具体地说会生成主菜单对象的句柄值。

```
hmenu=uimenu(hfig,'属性名', 属性值, …);
生成命令或子菜单对象的调用形式为:
hsub=uimenu(hfig,'属性名','属性值'…);
```

  这两种形式的区别在于：创建主菜单时，要给出图形窗口的句柄值，如果省略了这个句柄值，MATLAB 就在当前的图形窗口中生成这个主菜单。如果此时不存在当前的图形窗口，MATLAB 会自动打开一个图形窗口，并将该菜单作为主菜单对象。在创建命令时，指定父菜单对象的句柄值。

  其他属性的设置方式与控件对象属性的设置方式相同。

(1) 创建主菜单对象。

使用上述第一种调用形式创建主菜单对象时，要注意返回一个句柄值，以备后面定义命令或子菜单对象时使用。例如，为了定义一个名为 Plot Options 的主菜单，可以使用以下语句：

```
plotopt=uimenu(gcf,'label','plot' options);
```

说明： Label 属性值就是显示在图形窗口菜单栏中的主菜单的标题或菜单对象的名字，plotopt 是该主菜单对象的句柄值，供定义命令或子菜单对象使用。

(2) 创建命令。

使用上述第 2 种调用形式创建命令时，必须提供对应的父菜单对象的句柄值，作为该函数的第一个输入参数。如果命令又是一个子菜单对象或其他的语句要引用，则应该记录下 uimenu 函数的返回的句柄值。

下面的例子定义一个子菜单 Line Types 作为子菜单的命令。该父菜单对象的句柄值为 plotlt，各子菜单的 Callback 的工作是设置变量 ltype 之值：

```
plotlt=uimenu(plotopt,'Label','Line Types');
solid=uimenu(plotlt,'Label','Solid line',…
          'Callback','ltype='-';');
dotted=uimenu(plotlt,'Label','Dotted line',…
          'Callback','ltype=':';');
dashed=uimenu(plotlt,'Label','Dashed line',…
          'Callback','ltype='——';');
```

(3) 创建子菜单对象。

在创建子菜单对象的 uimenu 函数语句中，必须指明父菜单的句柄值，并返回子菜单对象的句柄值，供定义子菜单对象的的命令之用。下面的例子定义菜单对象 Plot Options 的 3 个命令，每一个命令都是子菜单，用分隔条将每个子菜单分开。

```
plotlt=uimenu(plotopt,'Label','Line Types');
plotsym=uimenu(plotopt,'Label','Plot Symbol','Separator',0'on');
plotcol=uimenu(plotopt,'Label','Plot Color','Separator','on');
```

(4) 创建带检测标记的命令对象。

MATLAB 对命令对象规定了一个属性 Checked，其属性值可以是 on 或 off，由此定义命令对象的两种没的状态。当命令的状态为 on 时，该命令名字的左端具有一个检测的标记，可利用这个标记指示出对该命令所作出的选择。

例如，下面的语句使得当用户选择 Solid Line 命令时，就在该命令的左端作一个标记。由于一次只能选择一个命令，因此，标记只出现在一个命令上。

```
solid=uimenu(plotlt,'Lable','Solid line',…
'Callback',['ltype='-';'…
'set(solid,'Checked','on',',…
'set(dotted,'Checked','off',',…
'set(dashed,'Checked','off')']);
```

(5) 创建切换式命令对象。

MATLAB 可以让创建的命令对象在两种状态之间进行切换，而命令的不同状态是通过命令的不同标题来标识的。

下面的例子定义的命令具有 on 和 off 两种状态，由命令的标题的改换来表明命令所处的状态。两种状态的标题分别是 on now 和 off now。当命令被选择时，菜单的名字就会改变，可以根据不同的状态来调用不同的子程序。

```
Toggle=uimenu(uimenu,'Lable','On Now',…
        'Callback',[if strcmp(get(toggle,'Lable'),'…
      'On Now'),',…
      'set(toggle,'Lable','Off Now'),',…
      'fnoff,',…
     'else,',…
      'set(toggle,'Label','On Now',', …
'fnon,',…
     'end']);
```

- 基于GUI的方式

  菜单编辑器的界面如图10.14所示，它是通过单击GUI设计板中的菜单编辑器(Menu Editor)按钮产生的。使用菜单编辑器，既可以在某个图形窗口的菜单条上添加用户自定义的菜单，也可以编辑已定义的某个用户菜单。下面举一个添加用户自定义菜单的例子。

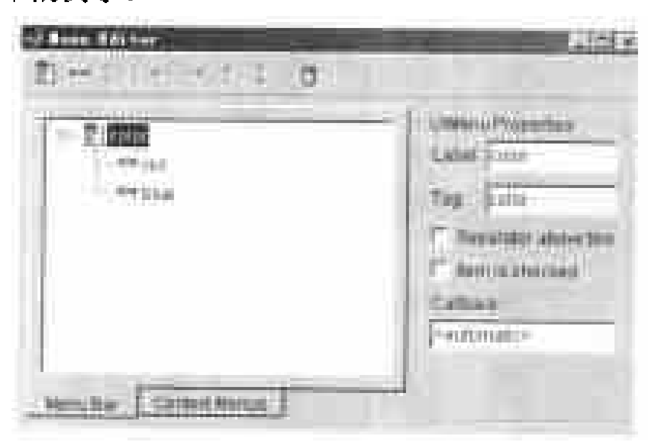


图 10.14 菜单编辑器

本例自制一个带下拉菜单表的用户菜单(如图10.15所示)，该菜单能使图形窗背景颜色设置为兰色或红色。

若用命令行方式，命令代码如下所示：

```
figure                                              %创建一个图形窗。
h_menu=uimenu(gcf,'label','Color');                 %制作用户顶层菜单项Color。
h_submenu1=uimenu(h_menu,'label','Blue',...         %制作下拉菜单项Blue。
    'callback','set(gcf,''Color'',''blue'')');
    %<4>
h_submenu2=uimenu(h_menu,'label','Red',... %制作下拉菜单Red。

    'callback','set(gcf,''Color'',''red'')');
```

结果如图10.16所示。

若用GUI方式，首先单击图10.15菜单编辑器中的第一个按钮，窗口中出现Untitled项，选中它，同时窗口右侧的3个可编辑文本框和两个复选框被激活，如图10.16所示。在Label和Tag文本框中输入“Color”。单击窗口中的第2个按钮，产生color菜单的第一个命令，同理在Label和Tag文本框中输入Blue和Red，在Callback文本框中输入上述第4行命令代码中的Callback属性值；照此法生成Red命令。其次，在GUI设计板属性浏览编辑器中，将MenuBar属性设为figure。最后激活刚设计好的GUI，生成图10.15所示的图形窗口，这样就可以通过Color菜单来改变窗口的颜色。

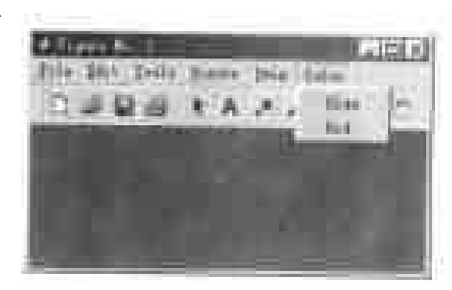


图 10.15 创建用户菜单示例

注意： 使用菜单编辑器，只能看到菜单的Label、Tag和Callback这3个属性。如果要对别的属性进行修改，则可以使用属性浏览编辑器。

图 10.16 用GUI方式创建用户菜单示例

2. 菜单对象的属性

- 公共属性
  - ◆ Children 属性

    Children 属性的取值为空矩阵或句柄值向量，因为菜单对象可以有自己的子菜单对象。对于命令来说，Children 的取值就是空矩阵。改变向量元素的顺序，可以改变菜单对象的子菜单及命令的顺序。
  - ◆ Parent 属性

    Parent 属性的取值是对象句柄值，因为菜单对象总是图形窗口对象的子菜单对象或另一个菜单对象的子菜单。该句柄值表明了菜单对象所在的图形窗口或其父菜单。如果用 set 函数将 Parent 属性值设置为另一个图形窗口的句柄值，则相当于将菜单对象转移到另一个图形窗口中。
  - ◆ Tag 属性

    Tag 属性的取值总是 uimenu，这个属性值表明图形对象的类型。菜单对象的类型就是 uimenu，用户不能改写这个属性。
  - ◆ UserData 属性

    UserData 属性的取值是一个矩阵，默认值为空矩阵。用户可以在这个属性中保存与该菜单对象相关的重要数据或信息，借此可以达到传递数据或信息的目的。用 set 和 get 函数可以访问该属性。
  - ◆ Visible 属性

    Visible 属性的取值是 on(默认值)或 off。如果该属性值为 off，那么菜单对象是不可见的，但是该菜单对象仍存在，用 set 和 get 函数可以访问这个菜单对象的每个属性。
- 基本控制属性
  - ◆ Accelerator 属性

    Accelerator 属性的取值是字符，只 Children 属性值为空的对象，即命令对象才有定义。它定义该命令的热键。在 Windows 环境下，用 Ctrl 键加该字符，即可激活该命令的 Callback 功能。

- ◆ Callback属性

  Callback属性的取值是字符串，可以是某个M文件名或一小段MATLAB语句。当用户激活菜单对象时，如果菜单对象没有子菜单，就运行Callback属性定义的子程序(Callback调用)。如果是子菜单对象，那么先运行Callback属性定义的子程序，再显示子菜单对象的子项。
- ◆ Checked属性

  Checked属性的取值是on或off(默认)，该属性为命令定义一个指示标记，可以用这个特性指明某种选择状态。
- ◆ Enable属性

  Enable属性的取值是on(默认值)或off，这个属性控制菜单对象的可选择性。如果它的值是off，则此时不能使用该菜单。此时，该菜单标题的外观是阴影。
- ◆ Lable属性

  Lable属性的取值是字符串，它定义菜单对象的名字。在字符串中加入&字符本身，字符并不出现在标题中。对于这种有带下划线字符的菜单，可以用Alt键加该字符键来激活。
- ◆ Position属性

  Position属性的取值是数值，它定义主菜单对象在菜单栏上的相对位置，或子菜单对象与命令对象在菜单组内的相对位置。例如，对主菜单，Position属性值为1，表示该菜单位于图形窗口菜单栏的可用位置的最左端。
- ◆ Scparator属性

  Separator属性的取值是on或off(默认值)。如果该属性值为on，则在该菜单项上方添加一条分隔线，用分隔线可以将各菜单按功能分开。

● Callback管理属性

- ◆ BusyAction属性

  BusyAction属性的取值是cancel或pueue(默认值)，该属性决定MATLAB采取的控制中断执行菜单对象的Callback调用方式。在MATLAB环境中，如果有一个Callback调用在运行，那么随后的Callback调用总是试图中断正在运行的Callback调用。如果此时Interruptible属性值为on，那么MATLAB在处理事件队列时，中断正在运行的Callback调用；如果Interruptible属性值为off，那么BusyAction属性就决定MATLAB处理事件的方式。如果它的值是cancel，就从事件队列中取消企图运行第2个Callback调用的事件；如果它的值是pueue，就将企图运行第二个Callback调用的事件加入事件队列，直到当前的Callback调用完成为止。
- ◆ CreateFcn属性

  CreateFcn属性的取值是字符串，一般是某个M文件名或一小段MATLAB语句。当MATLAB生成菜单对象时，即MATLAB事先应该执行的程序段。这个属性只能按设置默认值的方式定义，例如，下列语句

```
set(0,'Defaultuicontrolcreatefcn',……)
set(gcf,'IntegerHandle','off')');
```

从根对象层设置菜单对象的默认值。对于已经存在的菜单对象，改变该属性的值，对该对象无任何影响。MATLAB 生成菜单对象，完成所有默认属性值的设置后，再执行 CreateFcn 属性定义的 Callback。

如果一个菜单对象 CreateFcn 定义的 Callback 正在运行，那么该菜单对象的句柄值不可访问，必须用函数 gcbo 获取。

- HandleVisibility 属性

  HandleVisibility 属性的取值为 on(默认值)、Callback 或 off。这个属性定义菜单对象句柄的可访问权限，决定其句柄值是否在父对象的 Children 属性中出现。如果 HandleVisibility 属性值是 on，那么它的句柄值总是可见的。如果属性值为 Callback，那么该句柄值只在该对象的 Callback 调用以及 Callback 调用的函数内是可见的，但是命令行内调用的函数则不能访问这样的菜单句柄值。也就是说，这样的句柄对于该菜单对象自己的 Callback 调用是私有的。这对于保护交互式的程序是十分有用的。如果 HandleVisibility 属性值被设置为 off，那么菜单对象的句柄值在任何时刻都是不可见的。如果句柄是不可见的，那么任何查询句柄的函数都不能返回它的值。这些函数包括 get、findobj、gcf、gca、gco、newplot、cla、clf 及 close 等。当 HandleVisibility 属性值是 callback 或 off 时，这样的菜单对象不会出现在图形窗口对象的 Children 和 CurrentObject 属性中，也不会出现在根对象 CurrentObject 属性中。但是可以将根对象的 ShowHiddenHandles 属性设置为 on，临时使所有句柄都是可见的，而不管其 HandleVisibility 属性值是什么。被隐藏的句柄仍是有效的，如果知道这样的句柄值，则一切关于句柄的操作都是合法的。

- Interruptible 属性

  Interruptible 属性的取值是 on 或 off(默认值)，这个属性决定着菜单对象的 callback 是否可以随后的 callback 调用中断。只有由 ButtonDownFcn 和 callback 属性定义的 callback 受 Interruptible 属性值的影响。在运行程序时，只有遇到 drawnow、figure、getframe 和 pause 等命令时，MATLAB 系统才检查可以中断程序运行的 Callback 调用事件。

3. 菜单对象属性的修改

- 基于命令行的方式

  方法与前面相同，这里不再重复，总之，菜单对象作为 GUI 编程的一个基本构件，具有和其他 GUI 构件一样的控制方法。

- 基于 GUI 的方式

  如前所述，启动属性编辑器可以用前面介绍过的方法来进行修改。

附录 A MATLAB 图像处理工具箱函数

图像显示

函　　数	功　　能	语　　法
colorbar	显示颜色条	colorbar('vert') colorbar('horiz') colorbar(h) colorbar h = colorbar(...)
getimage	从坐标轴取得图像数据	A = getimage(h) [x,y,A] = getimage(h) [...,A,flag] = getimage(h) [...] = getimage
imshow	显示图像	imshow(I,n) imshow(I,[low high]) imshow(BW) imshow(X,map) imshow(RGB) imshow(...,display_option) imshow(x,y,A,...) imshow filename h = imshow(...)
montage	在矩形框中同时显示多幅图像	montage(I) montage(BW) montage(X,map) montage(RGB) h = montage(...)
immovie	创建多帧索引图的电影动画	mov = immovie(X,map)

续表

函　数	功　能	语　法
subimage	在一幅图中显示多个图像	subimage(X,map) subimage(I) subimage(BW) subimage(RGB) subimage(x,y,...) h = subimage(...)
truesize	调整图像显示尺寸	truesize(fig,[mrows mcols]) truesize(fig)
warp	将图像显示到纹理映射表面	warp(X,map) warp(I,n) warp(BW) warp(RGB) warp(z,...) warp(x,y,z,...) h = warp(...)
zoom	缩放图像	zoom on zoom off zoom out zoom reset zoom zoom xon zoom yon zoom(factor) zoom(fig,option)

图像文件 I/O

函　数	功　能	语　法
imfinfo	返回图形文件信息	info = imfinfo(filename,fmt) info = imfinfo(filename)
imread	从图形文件中读取图像	A = imread(filename,fmt) [X,map] = imread(filename,fmt) [...] = imread(filename) [...] = imread(...,idx) (TIFF only) [...] = imread(...,ref) (HDF only) [...] = imread(...,'BackgroundColor',BG)

续表

函 数	功 能	语 法
		(PNG only) [A,map,alpha] = imread(...) (PNG only)
imwrite	把图像写入图形文件中	imwrite(A,filename,fmt) imwrite(X,map,filename,fmt) imwrite(...,filename) imwrite(...,Param1,Val1,Param2,Val2...)

几何操作

函 数	功 能	语 法
imcrop	剪切图像	I2 = imcrop(I) X2 = imcrop(X,map) RGB2 = imcrop(RGB) I2 = imcrop(I,rect) X2 = imcrop(X,map,rect) RGB2 = imcrop(RGB,rect) [...] = imcrop(x,y,...) [A,rect] = imcrop(...) [x,y,A,rect] = imcrop(...)
imresize	改变图像大小	B = imresize(A,m,method) B = imresize(A,[mrows ncols],method) B = imresize(...,method,n) B = imresize(...,method,h)
imrotate	旋转图像	B = imrotate(A,angle,method) B = imrotate(A,angle,method,'crop')

像素和统计处理

函 数	功 能	语 法
corr2	计算两个矩阵的二维相关系数	r = corr2(A,B)
imcontour	创建图像数据的轮廓图	imcontour(I,n) imcontour(I,v) imcontour(x,y,...) imcontour(...,LineSpec) [C,h] = imcontour(...)

续表

函　数	功　能	语　法
imfeature	计算图像区域的特征尺寸	stats = imfeature(L,measurements) stats = imfeature(L,measurements,n)
imhist	显示图像数据的柱状图	imhist(I,n) imhist(X,map) [counts,x] = imhist(...)
impixel	确定像素颜色值	P = impixel(I) P = impixel(X,map) P = impixel(RGB) P = impixel(I,c,r) P = impixel(X,map,c,r) P = impixel(RGB,c,r) [c,r,P] = impixel(...) P = impixel(x,y,I,xi,yi) P = impixel(x,y,X,map,xi,yi) P = impixel(x,y,RGB,xi,yi) [xi,yi,P] = impixel(x,y,...)
improfile	沿线段计算剖面图的像素值	c = improfile c = improfile(n) c = improfile(I,xi,yi) c = improfile(I,xi,yi,n) [cx,cy,c] = improfile(...) [cx,cy,c,xi,yi] = improfile(...) [...] = improfile(x,y,I,xi,yi) [...] = improfile(x,y,I,xi,yi,n) [...] = improfile(...,method)
mean2	计算矩阵元素的平均值	b = mean2(A)
pixval	显示图像像素信息	pixval on pixval off pixval pixval(fig,option)
std2	计算矩阵元素的标准偏移	b = std2(A)

图像分析

函　　数	功　　能	语　　法
edge	识别强度图像中的边界	BW = edge(I,'sobel') BW = edge(I,'sobel',thresh) BW = edge(I,'sobel',thresh,direction) [BW,thresh] = edge(I,'sobel',...) BW = edge(I,'prewitt') BW = edge(I,'prewitt',thresh) BW = edge(I,'prewitt',thresh,direction) [BW,thresh] = edge(I,'prewitt',...) BW = edge(I,'roberts') BW = edge(I,'roberts',thresh) [BW,thresh] = edge(I,'roberts',...) BW = edge(I,'log') BW = edge(I,'log',thresh) BW = edge(I,'log',thresh,sigma) [BW,threshold] = edge(I,'log',...) BW = edge(I,'zerocross',thresh,h) [BW,thresh] = edge(I,'zerocross',...) BW = edge(I,'canny') BW = edge(I,'canny',thresh) BW = edge(I,'canny',thresh,sigma) [BW,threshold] = edge(I,'canny',...)
qtdecomp	进行四叉树分解	S = qtdecomp(I) S = qtdecomp(I,threshold) S = qtdecomp(I,threshold,mindim) S = qtdecomp(I,threshold,[mindim maxdim]) S = qtdecomp(I,fun) S = qtdecomp(I,fun,P1,P2,...)
qtgetblk	获取四叉树分解中的块值	[vals,r,c] = qtgetblk(I,S,dim) [vals,idx] = qtgetblk(I,S,dim)
qtsetblk	设置四叉树分解中的块值	J = qtsetblk(I,S,dim,vals)

图像加强

函　数	功　能	语　法
histeq	用柱状图均等化增强对比	J = histeq(I,hgram) J = histeq(I,n) [J,T] = histeq(I,...)
imadjust	调整图像灰度值或颜色映像表	J = imadjust(I,[low high],[bottom top],gamma) newmap = imadjust(map,[low high],[bottom top],gamma) RGB2 = imadjust(RGB1,...)
imnoise	增加图像的渲染效果	J = imnoise(I,type) J = imnoise(I,type,parameters)
medfilt2	进行二维中值过滤	B = medfilt2(A,[m n]) B = medfilt2(A) B = medfilt2(A,'indexed',...)
ordfilt2	进行二维统计顺序过滤	B = ordfilt2(A,order,domain) B = ordfilt2(A,order,domain,S) B = ordfilt2(...,padopt)
wiener2	进行二维适应性去噪过滤处理	J = wiener2(I,[m n],noise) [J,noise] = wiener2(I,[m n])

线性滤波

函　数	功　能	语　法
conv2	进行二维卷积操作	C = conv2(A,B) C = conv2(hcol,hrow,A) C = conv2(...,shape)
convmtx2	计算二维卷积矩阵	T = convmtx2(H,m,n) T = convmtx2(H,[m n])
convn	计算 n 维卷积	C = convn(A,B) C = convn(A,B,shape)
filter2	进行二维线性过滤操作	B = filter2(h,A) B = filter2(h,A,shape)
fspecial	创建预定义过滤器	h = fspecial(type) h = fspecial(type,parameters)

线性二维滤波设计

函　数	功　能	语　法
freqspace	确定二维频率响应的频率空间	[f1,f2] = freqspace(n) [f1,f2] = freqspace([m n]) [x1,y1] = freqspace(...,'meshgrid') f = freqspace(N) f = freqspace(N,'whole')
freqz2	计算二维频率响应	[H,f1,f2] = freqz2(h,n1,n2) [H,f1,f2] = freqz2(h,[n2 n1]) [H,f1,f2] = freqz2(h,f1,f2) [H,f1,f2] = freqz2(h) [...] = freqz2(h,...,[dx dy]) [...] = freqz2(h,...,dx) freqz2(...)
fsamp2	用频率采样法设计二维 FIR 过滤器	h = fsamp2(Hd) h = fsamp2(f1,f2,Hd,[m n])
ftrans2	通过频率转换设计二维 FIR 过滤器	h = ftrans2(b,t) h = ftrans2(b)
fwind1	用一维窗口方法设计二维 FIR 过滤器	h = fwind1(Hd,win) h = fwind1(Hd,win1,win2) h = fwind1(f1,f2,Hd,...)
fwind2	用二维窗口方法设计二维 FIR 过滤器	h = fwind2(Hd,win) h = fwind2(f1,f2,Hd,win)

图像变换

函　数	功　能	语　法
dct2	进行二维离散余弦变换	B = dct2(A) B = dct2(A,m,n) B = dct2(A,[m n])
dctmtx	计算离散余弦变换矩阵	D = dctmtx(n)
fft2	进行二维快速傅里叶变换	B = fft2(A) B = fft2(A,m,n)
fftn	进行 n 维快速傅里叶变换	B = fftn(A) B = fftn(A,siz)

续表

函　数	功　能	语　法
fftshift	把快速傅里叶变换的 DC 组件移到光谱中心	B = fftshift(A)
idct2	计算二维离散反余弦变换	B = idct2(A) B = idct2(A,m,n) B = idct2(A,[m n])
ifft2	计算二维快速傅里叶反变换	B = ifft2(A) B = ifft2(A,m,n)
ifftn	计算 n 维快速傅里叶反变换	B = ifftn(A) B = ifftn(A,siz)
iradon	进行反 radon 变换	I = iradon(P,theta) I = iradon(P,theta,interp,filter,d,n) [I,h] = iradon(...)
phantom	产生一个头部幻影图像	P = phantom(def,n) P = phantom(E,n) [P,E] = phantom(...)
radon	计算 radon 变换	R = radon(I,theta) R = radon(I,theta,n) [R,xp] = radon(...)

边沿和块处理

函　数	功　能	语　法
bestblk	确定进行块操作的块大小	siz = bestblk([m n],k) [mb,nb] = bestblk([m n],k)
blkproc	实现图像的显式块操作	B = blkproc(A,[m n],fun) B = blkproc(A,[m n],fun,P1,P2,...) B = blkproc(A,[m n],[mborder nborder],fun,...) B = blkproc(A,'indexed',...)
col2im	将矩阵的列重新组织到块中	A = col2im(B,[m n],[mm nn],block_type) A = col2im(B,[m n],[mm nn])
colfilt	利用列相关函数进行边沿操作	B = colfilt(A,[m n],block_type,fun) B = colfilt(A,[m n],block_type,fun,P1,P2,...) B = colfilt(A,[m n],[mblock nblock],block_type,fun,...) B = colfilt(A,'indexed',...)

续表

函　数	功　能	语　法
im2col	重调图像块为列	B = im2col(A,[m n],block_type) B = im2col(A,[m n]) B = im2col(A,'indexed',...)
nlfilter	进行边沿操作	B = nlfilter(A,[m n],fun) B = nlfilter(A,[m n],fun,P1,P2,...) B = nlfilter(A,'indexed',...)

二进制图像操作

函　数	功　能	语　法
applylut	在二进制图像中利用 lookup 表进行边沿操作	A = applylut(BW,lut)
bwarea	计算二进制图像对象的面积	total = bwarea(BW)
bweuler	计算二进制图像的欧拉数	eul = bweuler(BW,n)
bwfill	填充二进制图像的背景色	BW2 = bwfill(BW1,c,r,n) BW2 = bwfill(BW1,n) [BW2,idx] = bwfill(...) BW2 = bwfill(x,y,BW1,xi,yi,n) [x,y,BW2,idx,xi,yi] = bwfill(...) BW2 = bwfill(BW1,'holes',n) [BW2,idx] = bwfill(BW1,'holes',n)
bwlabel	标注二进制图像中已连接的部分	L = bwlabel(BW,n) [L,num] = bwlabel(BW,n)
bwmorph	提取二进制图像的轮廓	BW2 = bwmorph(BW1,operation) BW2 = bwmorph(BW1,operation,n)
bwperim	计算二进制图像中对象的周长	BW2 = bwperim(BW1,n)
bwselect	在二进制图像中选择对象	BW2 = bwselect(BW1,c,r,n) BW2 = bwselect(BW1,n) [BW2,idx] = bwselect(...)
dilate	放大二进制图像	BW2 = dilate(BW1,SE) BW2 = dilate(BW1,SE,alg) BW2 = dilate(BW1,SE,...,n)
erode	弱化二进制图像的边界	BW2 = erode(BW1,SE) BW2 = erode(BW1,SE,alg) BW2 = erode(BW1,SE,...,n)

续表

函　　数	功　　能	语　　法
makelut	创建一个用于 applylut 函数的 lookup 表	lut = makelut(fun,n) lut = makelut(fun,n,P1,P2,...)

区域处理

函　　数	功　　能	语　　法
roicolor	选择感兴趣的颜色区	BW = roicolor(A,low,high) BW = roicolor(A,v)
roifill	在图像的任意区域中进行平滑插补	J = roifill(I,c,r) J = roifill(I) J = roifill(I,BW) [J,BW] = roifill(...) J = roifill(x,y,I,xi,yi) [x,y,J,BW,xi,yi] = roifill(...)
roifilt2	过滤敏感区域	J = roifilt2(h,I,BW) J = roifilt2(I,BW,fun) J = roifilt2(I,BW,fun,P1,P2,...)
roipoly	选择一个敏感的多边形区域	BW = roipoly(I,c,r) BW = roipoly(I) BW = roipoly(x,y,I,xi,yi) [BW,xi,yi] = roipoly(...) [x,y,BW,xi,yi] = roipoly(...)

颜色映像处理

函　　数	功　　能	语　　法
brighten	增加或降低颜色映像表的亮度	brighten(beta) newmap = brighten(beta) newmap = brighten(map,beta) brighten(fig,beta)
cmpermute	调整颜色映像表中的颜色	[Y,newmap] = cmpermute(X,map) [Y,newmap] = cmpermute(X,map,index)

续表

函　　数	功　　能	语　　法
cmunique	查找颜色映象表中特定的颜色及相应的图像	[Y,newmap] = cmunique(X,map) [Y,newmap] = cmunique(RGB) [Y,newmap] = cmunique(I)
imapprox	对索引图像进行近似处理	[Y,newmap] = imapprox(X,map,n) [Y,newmap] = imapprox(X,map,tol) Y = imapprox(X,map,newmap) [...] = imapprox(...,dither_option)
rgbplot	划分颜色映像表	rgbplot(map)

颜色空间转换

函　　数	功　　能	语　　法
hsv2rgb	转换 HSV 值为 RGB 颜色空间	rgbmap = hsv2rgb(hsvmap) RGB = hsv2rgb(HSV)
ntsc2rgb	转换 NTSC 的值为 RGB 颜色空间	rgbmap = ntsc2rgb(yiqmap) RGB = ntsc2rgb(YIQ)
rgb2hsv	转化 RGB 值为 HSV 颜色空间	hsvmap = rgb2hsv(rgbmap) HSV = rgb2hsv(RGB)
rgb2ntsc	转化 RGB 的值为 NTSC 颜色空间	yiqmap = rgb2ntsc(rgbmap) YIQ = rgb2ntsc(RGB)
rgb2ycbcr	转化 RGB 的值为 YcbCr 颜色空间	ycbcrmap = rgb2ycbcr(rgbmap) YCBCR = rgb2ycbcr(RGB)
ycbcr2rgb	转化 YcbCr 值为 RGB 颜色空间	rgbmap = ycbcr2rgb(ycbcrmap) RGB = ycbcr2rgb(YCBCR)

图像类型和类型转换

函　　数	功　　能	语　　法
dither	通过抖动增加外观颜色分辨率，转换图像	X = dither(RGB,map) BW = dither(I)
gray2ind	转换灰度图像为索引图像	[X,map] = gray2ind(I,n)
grayslice	从灰度图像创建索引图像	X = grayslice(I,n) X = grayslice(I,v)

续表

函　数	功　能	语　法
im2bw	转换图像为二进制图像	BW = im2bw(I,level) BW = im2bw(X,map,level) BW = im2bw(RGB,level)
im2double	转换图像矩阵为双精度型	I2 = im2double(I1) RGB2 = im2double(RGB1) BW2 = im2double(BW1) X2 = im2double(X1,'indexed')
double	转换数据为双精度型	B = double(A)
uint8	转换数据为 8 位无符号整型	B = uint8(A)
im2uint8	转换图像阵列为 8 位无符号整型	I2 = im2uint8(I1) RGB2 = im2uint8(RGB1) BW2 = im2uint8(BW1) X2 = im2uint8(X1,'indexed')
im2uint16	转换图像阵列为 16 位无符号整型	I2 = im2uint16(I1) RGB2 = im2uint16(RGB1) X2 = im2uint16(X1,'indexed')
uint16	转换数据为 16 位无符号整型	I = uint16(X)
ind2gray	把检索图像转化为灰度图像	I = ind2gray(X,map)
ind2rgb	转化索引图像为 RGB 真彩图像	RGB = ind2rgb(X,map)
isbw	判断是否为二进制图像	flag = isbw(A)
isgray	判断是否为灰度图像	flag = isgray(A)
isind	判断是否为索引图像	flag = isind(A)
isrgb	判读是否为 RGB 真彩图像	flag = isrgb(A)
mat2gray	转化矩阵为灰度图像	I = mat2gray(A,[amin amax]) I = mat2gray(A)
rgb2gray	转换 RGB 图像或颜色映像表为灰度图像	I = rgb2gray(RGB) newmap = rgb2gray(map)
rgb2ind	转化 RGB 图像为索引图像	[X,map] = rgb2ind(RGB,tol) [X,map] = rgb2ind(RGB,n) X = rgb2ind(RGB,map) [...] = rgb2ind(...,dither_option)

工具箱参数设置

函　数	功　能	语　法
iptgetpref	获取图像处理工具箱参数设置	value = iptgetpref(prefname)
iptsetpref	设置图像处理工具箱参数	iptsetpref(prefname,value)

附录 B　MATLAB 小波分析工具箱函数

一般问题

函数名	功　能	语　法
biorfilt	产生双正交小波滤波器组	[Lo_D,Hi_D,Lo_R,Hi_R] = biorfilt(DF,RF) [Lo_D1,Hi_D1,Lo_R1,Hi_R1,Lo_D2,Hi_D2,Lo_R2,Hi_R2] =biorfilt(DF,RF,'8')
centfrq	计算小波中心频率	FREQ = centfrq('wname') FREQ = centfrq('wname',ITER) [FREQ,XVAL,RECFREQ]= centfrq('wname', ITER, 'plot')
dyaddown	信号的二倍抽取	Y = dyaddown(X,EVENODD) Y = dyaddown(X) Y = dyaddown(X,EVENODD,'type') Y = dyaddown(X,'type',EVENODD)
dyadup	信号的二倍补零插值	Y = dyadup(X,EVENODD) Y = dyadup(X) Y = dyadup(X,EVENODD,'type') Y = dyadup(X,'type',EVENODD)
intwave	计算小波的积分函数	[INTEG,XVAL] = intwave('wname',PREC) [INTEG,XVAL] = intwave('wname',PREC,PFLAG) [INTEG,XVAL] = intwave('wname')
orthfilt	产生正交小波滤波器组	[Lo_D,Hi_D,Lo_R,Hi_R] = orthfilt(W)
qmf	构造二次镜像滤波器组	Y = qmf(X,P) Y = qmf(X)
scal2frq	尺度到频率的变换	F = scal2frq(A,'wname',DELTA)
wavefun	产生小波函数和尺度函数	[PHI,PSI,XVAL] = wavefun('wname',ITER) [PHI1,PSI1,PHI2,PSI2,XVAL] = wavefun('wname',ITER) [PSI,XVAL] = wavefun('wname',ITER) wavefun ('wname',A,B)

续表

函数名	功　能	语　法
wavemngr	小波管理器，用于添加、删除、恢复或者读取小波	wavemngr('add',FN,FSN,WT,NUMS,FILE) wavemngr('add',FN,FSN,WT,NUMS,FILE,B) wavemngr('del',N) wavemngr('restore') wavemngr('restore',IN2) OUT1 = wavemngr('read') OUT1 = wavemngr('read',IN2) OUT1 = wavemngr('read_asc')
wfilters	由正交或者双正交的小波构造滤波器组	[Lo_D,Hi_D,Lo_R,Hi_R] = wfilters('wname') [F1,F2] = wfilters('wname','type')
wmaxlev	给出信号在给定小波情况下的最大分解层数	L = wmaxlev(S,'wname')

小波函数族

函数名	功　能	语　法
biorwavf	双正交样条小波滤波器	[RF,DF] = biorwavf(W)
cgauwavf	复数高斯小波	[PSI,X] = cgauwavf(LB,UB，N,P)
cmorwavf	复数 Morlet 小波	[PSI,X] = cmorwavf(LB,UB, N,FB,FC)
coifwavf	Coiflet 小波滤波器	F = coifwavf(W)
dbaux	计算 Daubechies 小波滤波器	W = dbaux(N,SUMW) W = dbaux(N)
dbwavf	Daubechies 小波滤波器	F = dbwavf(W)
fbspwavf	复频率B—样条小波滤波器	[PSI,X] = fbspwavf(LB,UB,N, M,FB, FC)
gauswavf	高斯小波	[PSI,X] = gauswavf(LB,UB, N,P)
mexihat	墨西哥草帽小波	[PSI,X] = mexihat(LB,UB,N)
meyer	Meyer 小波	[PHI,PSI,T] = meyer(LB,UB, N) [PHI,T] = meyer(LB,UB,N, 'phi') [PSI,T] = meyer(LB,UB,N, 'psi')
meyeraux	Meyer 小波辅助函数	Y = meyeraux(X)
morlet	Morlet 小波	[PSI,X] = morlet(LB,UB,N)
rbiowavf	逆双正交样条小波滤波器	[RF,DF]= rbiowavf(W)
shanwavf	复数 Shannon 小波	[PSI,X] = shanwavf(LB,UB, N,FB,FC)

续表

函数名	功　　能	语　　法
symaux	计算 Symlet 小波	W = SYMAUX(N,SUMW) W = SYMAUX(N)
symwavf	Symlet 小波滤波器	F = symwavf(W)

一维离散小波变换

函数名	功　　能	语　　法
appcoef	提取一维近似分量系数	A = appcoef(C,L,'wname',N) A = appcoef(C,L,'wname') A = appcoef(C,L,Lo_R, Hi_R) A = appcoef(C,L,Lo_R, Hi_R,N)
detcoef	提取一维细节分量系数	D = detcoef(C,L,N) D = detcoef(C,L)
dwt	一维离散小波变换	[cA,cD] = dwt(X,'wname') [cA,cD] = dwt(X,'wname', 'mode', MODE) [cA,cD] = dwt(X,Lo_D, Hi_D) [cA,cD] = dwt(X,Lo_D, Hi_D, 'mode', MODE)
dwtmode	设置一维离散小波变换模式	ST = dwtmode dwtmode('mode')
idwt	一维离散小波反变换	X = idwt(cA,cD,'wname') X = idwt(cA,cD,Lo_R,Hi_R) X = idwt(cA,cD,'wname',L) X = idwt(cA,cD,Lo_R, Hi_R,L) X = idwt(...,'mode',MODE)
upcoef	由一维离散小波变换求解近似和细节分量	Y = upcoef(O,X,'wname',N) Y = upcoef(O,X,'wname', N,L) Y = upcoef(O,X,Lo_R, Hi_R,N) Y = upcoef(O,X,Lo_R, Hi_R,N,L) Y = upcoef(O,X,'wname') Y = upcoef(O,X,Lo_R,Hi_R)
upwlev	单层小波分解的重构	[NC,NL,cA] = upwlev(C,L,'wname') [NC,NL,cA] = upwlev(C,L,Lo_R,Hi_R)
wavedec	多层一维小波分解	[C,L] = wavedec(X,N,'wname') [C,L] = wavedec(X,N, Lo_D, Hi_D)
waverec	多层一维小波重构	X = waverec(C,L,'wname') X = waverec(C,L, Lo_R, Hi_R)

续表

函数名	功　能	语　法
wrcoef	二维信号小波重构	X = wrcoef('type',C,L,'wname',N) X = wrcoef('type',C,L,Lo_R, Hi_R,N) X = wrcoef('type',C,L, 'wname') X = wrcoef('type',C,L,Lo_R, Hi_R)

二维离散小波变换

函数名	功　能	语　法
appcoef2	提取二维离散小波变换的近似分量	A = appcoef2(C,S,'wname',N) A = appcoef2(C,S,'wname') A = appcoef2(C,S,Lo_R,Hi_R) A = appcoef2(C,S,Lo_R,Hi_R,N)
detcoef2	提取二维离散小波变换的细节分量	D = detcoef(C,L,N) D = detcoef(C,L)
dwt2	单层二维离散小波变换	[cA,cH,cV,cD] = dwt2(X,'wname') [cA,cH,cV,cD] = dwt2(X,Lo_D,Hi_D)
dwtmode	设置二维离散小波变换的模式	ST = dwtmode dwtmode('mode')
idwt2	单层二维离散小波反变换	X = idwt2(cA,cH,cV,cD,'wname') X = idwt2(cA,cH,cV,cD, Lo_R,Hi_R) X = idwt2(cA,cH,cV,cD,'wname',S) X = idwt2(cA,cH,cV,cD, Lo_R,Hi_R,S) X = idwt2(...,'mode',MODE)
upcoef2	由多层小波分解重构近似分量或者细节分量	Y = upcoef2(O,X,'wname',N,S) Y = upcoef2(O,X,Lo_R, Hi_R,N,S) Y = upcoef2(O,X,'wname',N) Y = upcoef2(O,X,Lo_R,Hi_R,N) Y = upcoef2(O,X,'wname') Y = upcoef2(O,X,Lo_R,Hi_R)
upwlev2	二维信号小波分解的单层重构	[NC,NS,cA] = upwlev2(C,S, 'wname') [NC,NS,cA] = upwlev2(C,S, Lo_R, Hi_R)
wavedec2	二维信号的多层小波分解	[C,S] = wavedec2(X,N,'wname') [C,S] = wavedec2(X,N,Lo_D,Hi_D)
waverec2	二维信号小波的多层重构	X = waverec2(C,S,'wname') X = waverec2(C,S,Lo_R,Hi_R)

续表

函数名	功能	语法
wrcoef2	由多层小波分解重构某一层的分解信号	X = wrcoef2('type',C,S, 'wname',N) X = wrcoef2('type',C,S,Lo_R,Hi_R,N) X = wrcoef2('type',C,S,'wname') X=wrcoef2('type',C,S, Lo_R,Hi_R)

小波包算法

函数名	功能	语法
bestlevt	利用小波包分解寻找最优树	T = bestlevt(T) [T,E] = bestlevt(T)
besttree	利用小波包分解寻找最优层次树	T = besttree(T) [T,E] = besttree(T) [T,E,N] = besttree(T)
entrupd	按新的熵函数更新分解结构	T = entrupd(T,ENT) T = entrupd(T,ENT,PAR)
wentropy	计算熵	E = wentropy(X,T,P) E = wentropy(X,T)
wp2wtree	从小波包分解树中提取小波分解树	T = wp2wtree(T)
wpcoef	按指定节点的小波包分解	X = wpcoef(T,N) X = wpcoef(T)
wpcutree	对小波包分解树进行剪切	T = wpcutree(T,L) [T,RN] = wpcutree(T,L)
wpdec	小波包的一维分解	T = wpdec(X,N,'wname',E,P) T = wpdec(X,N,'wname')
wpdec2	小波包的二维分解	T = wpdec2(X,N,'wname', E,P) T = wpdec2(X,N,'wname')
wpfun	小波包函数族	[WPWS,X] = wpfun('wname',NUM,PREC) [WPWS,X]= wpfun('wname',NUM)
wpjoin	重组小波包	T = wpjoin(T,N) [T,X] = wpjoin(T,N) T = wpjoin(T) [T,X] = wpjoin(T)
wprcoef	重构小波包分解树的系数	X = wprcoef(T,N)
wprec	小波包的一维重构	X = wprec(T)
wprec2	小波包的二维重构	X = wprec2(T)

续表

函数名	功　能	语　法
wpsplt	分裂小波包分解的终端节点	T = wpsplt(T,N) [T,cA,cD] = wpsplt(T,N) [T,cA,cH,cV,cD]= wpsplt(T,N)

小波去噪和压缩函数

函数名	功　能	语　法
ddencmp	自动生成小波去噪或者压缩的阈值选择方案	[THR,SORH,KEEPAPP,CRIT] = ddencmp(IN1,IN2,X) [THR,SORH,KEEPAPP] = ddencmp(IN1,'wv',X) [THR,SORH,KEEPAPP,CRIT] = ddencmp(IN1,'wp',X)
thselect	为利用小波分解去噪选取阈值	THR = thselect(X,TPTR)
wbmpen	根据惩罚函数对小波的一维或者二维去噪选择阈值	THR = wbmpen(C,L,SIGMA,ALPHA)
wdcbm	利用 Birge- Massart 准则进行一维小波去噪	[THR,NKEEP] = wdcbm(C,L,ALPHA) [THR,NKEEP] = wdcbm(C,L,ALPHA,M)
wdcbm2	利用 Birge- Massart 准则进行二维小波去噪	[THR,NKEEP] = wdcbm2(C,S,ALPHA) [THR,NKEEP] = wdcbm2(C,S,ALPHA,M)
wden	自动利用小波进行一维信号的去噪	[XD,CXD,LXD] = wden(X,TPTR,SORH,SCAL,N,'wname') [XD,CXD,LXD] = wden(C,L,TPTR,SORH,SCAL,N,'wname')
wdencmp	利用小波对信号进行去噪或者压缩	[XC,CXC,LXC,PERF0,PERFL2] =wdencmp('gbl',X,'wname',N,THR,SORH,KEEPAPP) [XC,CXC,LXC,PERF0,PERFL2] = wdencmp('lvd',X,'wname',N, THR,SORH) [XC,CXC,LXC,PERF0,PERFL2] = wdencmp('lvd',C,L,'wname',N, THR,SORH)
wnoise	为检验小波去噪性能产生测试噪声	X = wnoise(FUN,N) [X,XN] = wnoise(FUN,N,SQRT_SNR) [X,XN] = wnoise(FUN,N,SQRT_SNR,INIT)
wnoisest	估计一维小波系数中噪声的能量	STDC = wnoisest(C,L,S)

续表

函数名	功能	语法
wpbmpen	利用 Birge- Massart 准则进行小波包去噪	THR = wpbmpen(T,SIGMA,ALPHA) THR = wpbmpen(T,SIGMA,ALPHA,ARG)
wpdencmp	利用小波包分解进行信号的去噪或者压缩	[XD,TREED,PERF0,PERFL2] = wpdencmp(X,SORH,N,'wname',CRIT,PAR,KEEPAPP) [XD,TREED,PERF0,PERFL2] = wpdencmp(TREE,SORH,CRIT,PAR,KEEPAPP)
wpthcoef	对小波包分解系数进行阈值处理	T = wpthcoef(T,KEEPAPP,SORH,THR)
wthcoef	对一维小波分解系数进行阈值处理	NC = wthcoef('d',C,L,N,P) NC = wthcoef('d',C,L,N) NC = wthcoef('a',C,L) NC = wthcoef('t',C,L,N,T,SORH)
wthcoef2	对二维小波分解系数进行阈值处理	NC = wthcoef2('type',C,S,N,T,SORH) NC = wthcoef2('type',C,S,N) NC = wthcoef2('a',C,S) NC = wthcoef2('t',C,S,N,T,SORH)
wthresh	进行软阈值或者硬阈值处理	Y = wthresh(X,SORH,T)
wthrmngr	阈值设置管理	THR = wthrmngr(OPTION,METHOD,VARARGIN)

小波包分解树管理函数

函数名	功能	语法
allnodes	列出树结构的所有节点	N = allnodes(T) N = allnodes(T,'deppos')
depo2ind	将深度/位置矩阵转化为顺序存储结构	N = depo2ind(ORD,[D P])
drawtree	画出小波包分解树的示意图	drawtree(T) drawtree(T,F) F = drawtree(T)
dtree	建立 dtree 类	T = dtree(ORD,D,X) T = dtree(ORD,D,X,U) [T,NB] = dtree(...) T = dtree('PropName1',PropValue1,'PropName2',PropValue2, ...)

续表

函数名	功　能	语　法
get	得到树对象的域的内容	[FieldValue1,FieldValue2, ...] = get(T,'FieldName1', 'Field- Name2', ...) [FieldValue1,FieldValue2, ...] = get(T)
ind2depo	把节点的索引号转化为节点的深度位置	[D,P] = ind2depo(ORD,N)
isnode	判断指定的位置上是否存在节点	R = isnode(T,N)
istnode	判断一个节点是否为终端节点	R = istnode(T,N)
leaves	判断是否为终端节点	N = leaves(T) N = leaves(T,'dp') [N,K] = leaves(T,'sort') [N,K] = leaves(T,'sortdp')
nodeasc	求指定节点按先序遍历的前驱	A = nodeasc(T,N) A = nodeasc(T,N,'deppos')
nodedesc	求指定节点按先序遍历的后继	D = nodedesc(T,N) D = nodedesc(T,N,'deppos')
nodejoin	树的剪枝	T = nodejoin(T,N) T = nodejoin(T)
nodepar	求指定节点在先序遍历下的父节点	F = nodepar(T,N) F = nodepar(T,N,'deppos')
nodesplt	分裂终端节点	T = nodesplt(T,N)
noleaves	判断是否为非终端节点	N = noleaves(T) N = noleaves(T,'dp')
ntnode	计算终端节点的数目	NB = ntnode(T)
ntree	产生 ntree 类	T = ntree(ORD,D) T = ntree T = ntree(ORD) T = ntree(ORD,D,S,U) T = ntree('PropName1',PropValue1,'PropName2', Prop Value2, ...)
plot	读取树对象的域的内容	plot(T) plot(T,FIG) FIG = plot(T) NEWT = plot(T,'read',FIG) NEWT = plot(DUMMY,'read',FIG)

续表

函数名	功能	语法
read	从一幅图中读取小波包分解树的结构	VARARGOUT = read(T,VARARGIN) 具体形式为: PropValue = read(T,'PropName') PropValue = read(T,'PropName','PropParam') [PropValue1,PropValue2, ...] = read(T,'PropName1','PropParam1','PropName2','PropParam2', ...)
readtree	画出一个树对象	T = readtree(F)
set	为树对象的域赋值	T = set(T,'FieldName1',FieldValue1,'FieldName2', FieldValue2, ...)
tnodes	判断是否为终端节点	N = tnodes(T) N = tnodes(T,'deppos') [N,K] = tnodes(T) [N,K] = tnodes(T,'deppos')
treedpth	计算树的深度	D = treedpth(T)
treeord	计算树的阶数	ORD = treeord(T)
wptree	建立 wptree 类	T = wptree(ORDER,DEPTH,X,WNAME,ENT_TYPE, ENT_PAR) T = wptree(ORDER,DEPTH,X,WNAME) T = wptree(... ,USERDATA)
wpviewcf	画出小波包分解树系数的彩色图	wpviewcf(T,CMODE) wpviewcf(T,CMODE,NBCOL)
write	给树对象的域赋值	T = write(T,'cfs',NODE,COEFS) T = write(T,'cfs',N1,CFS1,'cfs',N2,CFS2, ...)
wtbo	建立 wtbo 类	OBJ = wtbo OBJ = wtbo(USERDATA)
wtreemgr	NTREE 对象管理，根据 OPT 的实现树操作的功能	VARARGOUT = wtreemgr(OPT,T,VARARGIN) OPT 的可选值: 'allnodes' : Tree nodes 'isnode' : True for existing node 'istnode' : True for terminal nodes 'nodeasc' : Node ascendants 'nodedesc' : Node descendants 'nodepar' : Node parent 'ntnode' : Number of terminal nodes 'tnodes' : Terminal nodes

续表

函数名	功　能	语　法
		'leaves' : Terminal nodes 'noleaves' : Not terminal nodes 'order' : Tree order 'depth' : Tree depth

其他函数

函数名	功　能	语　法
wcodemat	对矩阵进行扩展的伪彩色编码	Y = wcodemat(X,NBCODES,OPT,ABSOL) Y = wcodemat(X,NBCODES,OPT) Y = wcodemat(X,NBCODES) Y = wcodemat(X)
wextend	扩展矢量或者矩阵	Y = wextend(TYPE,MODE,X,L,LOC) Y = wextend(TYPE,MODE,X,L)
wkeep	保留矢量或者矩阵的某一部分	Y = wkeep(X,L,OPT) Y = wkeep(X,L)
wrev	矢量翻转	Y = wrev(X)
instdfft	非标准一维傅立叶反变换	[X,T] = instdfft(XHAT,LOWB,UPPB)
nstdfft	非标准一维快速傅立叶变换	[XHAT,OMEGA] = nstdfft(X,LOWB,UPPB)
wvarchg	寻找方差变换点	[PTS_OPT,KOPT,T_EST] = wvarchg(Y,K,D) [PTS_OPT,KOPT,T_EST] = wvarchg(Y,K) [PTS_OPT,KOPT,T_EST] = wvarchg(Y)
waveinfo	显示小波函数信息	waveinfo waveinfo('wname')
wavedemo	小波分析演示程序	wavedemo
wavemenu	启动小波分析图像用户界面	wavemenu
cwt	一维连续小波变换(既可以是实数也可以是复数)	COEFS = cwt(S,SCALES,'wname') COEFS = cwt(S,SCALES,'wname','plot') COEFS = cwt(S,SCALES,'wname',PLOTMODE) COEFS=cwt(S,SCALES,'wname',PLOTMODE,XLIM)
iswt	一维离散平稳小波反变换	X = iswt(SWC,'wname') X = iswt(SWA,SWD,'wname') X = iswt(SWC,Lo_R,Hi_R) X = iswt(SWA,SWD,Lo_R,Hi_R)

续表

函数名	功能	语法
iswt2	二维离散平稳小波反变换	X = iswt2(SWC,'wname') X = iswt2(A,H,V,D,'wname') X = iswt2(SWC,Lo_R,Hi_R) X = iswt2(A,H,V,D,Lo_R,Hi_R)
swt	一维离散平稳小波变换	SWC = swt(X,N,'wname') SWC = swt(X,N,Lo_D,Hi_D) [SWA,SWD] = swt(X,N,'wname') [SWA,SWD] = swt(X,N,Lo_D,Hi_D)
swt2	二维离散平稳小波变换	SWC = swt2(X,N,'wname') [A,H,V,D] = swt2(X,N,'wname') SWC = swt2(X,N,Lo_D,Hi_D) [A,H,V,D] = swt2(X,N,Lo_D,Hi_D)